ESD-TR-71-383

MTR-2203

AIDS USERS' MANUAL

C.A. Marcus

AUGUST 1971

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 4060

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract F19(628)-71-C-0002

ESD-TR-71-383

MTR-2203

AIDS USERS' MANUAL

C.A. Marcus

AUGUST 1971

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 4060

Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts

Contract F19(628)-71-C-0002

FOREWORD

This document is a users' manual for a data management system designed for answering requests for specific information using Honeywell (GE) G-635 computers. This system was developed for the Data Services Center, Hq USAF, under Development Directive 79, dated 29 September 1970, and AFSC Program Direction (6917-13-71-28), dated 25 November 1970. The system development was accomplished by General Electric Co., Space Division under contract F19(628)-71-C-0181. This document was prepared by The MITRE Corporation under contract F19(628)-71-C-0002. Technical design and direction of this project is by Capt Frederick P. Ariail, Technology Applications Division (MCDT), Directorate of Systems Design and Development (MCD).

Developmental testing and evaluation of this system were provided by The MITRE Corporation and Rome Air Development Center, Information Sciences Division.

REVIEW AND APPROVAL

This technical report has been reviewed and approved.

EDMUND P. GAINES, JR., Colonel, USAF
Director, Systems Design & Development
Deputy for Command & Management Systems

ABSTRACT

AIDS is a computer software package designed to provide data management capabilities to a wide variety of users. It is written primarily in Common Business Oriented Language (COBOL) and is designed and implemented on the Honeywell G-635. This system, originally developed for NASA by General Electric Co., Apollo Systems Division, was named Manned Space Flight-Data Processing System (MSF-DPS). Modifications were made to improve the capabilities of MSF-DPS. These modifications, designed to meet interim requirements of the Air Force Data Services Center (AF/ACS), provide a responsive, versatile data management system for users of Honeywell (GE) 600 series computers.

This technical report, designed for users of AIDS, details the features of this system and provides examples of its use. Detailed system description (installation, maintenance, internal linkages, etc.) are not contained in this report. This information is contained in the AIDS Operations Manual (General Electric Co., "AIDS Version Description Document," July 1971). Organizations using AIDS may further modify or enhance this system independently of ESD; therefore, no attempt will be made by ESD to update this Technical Report if such changes are made.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

The AF/ACS Interim Data Management System (AIDS) is a computer software package designed to provide data management capabilities to a wide variety of users. It is written primarily in Common Business Oriented Language (COBOL) and is designed and implemented on the Honeywell G-635. This paper, in conjunction with the AIDS Operations Manual, provides the documentation needed to use the AIDS system.

AIDS SYSTEM CONCEPTS

AIDS is a data management system aimed at answering requests for specific information stored in formatted files. The system provides support in the following areas:

i. One time non-routine requests.

ii. Specific and selective information requests.

iii. Non-periodic, as well as periodic, reports.

iv. Definition of formats for periodic reports.

v. Surveys of file contents by users.

AIDS is best applied to production of unscheduled reports or requests for specific or selective information from available data files. AIDS can satisfy the requirements for ad hoc requests by eliminating the need for special programming to produce one-time reports. This implies that the types of requests most appropriate are those for which the output represents a small percentage of the total data file. Reports that the user may want to produce on a routine basis should probably be generated by using special-purpose COBOL programs.

AIDS accesses data files through a dictionary which describes the structure of each file to be searched. All data files must be maintained and generated externally to AIDS, usually by the individual disciplines requiring the data; i.e., procurement, management, cost analysis. AIDS can be used to search these files and to produce reports.

Definition of Terms

The following terms are defined in order to permit brevity in describing AIDS and to help eliminate confusion about each item:

Data Value is the basic unit of information which is not normally subdivided. This value will be a name, number, or combination of both.

Data Field is one or more characters conveying some type of information and containing a data value. Data fields within a record are searched to satisfy a requirement. Fields are given English titles for reference and are listed in the dictionary file.

Record is a collection of data fields that satisfy a particular function within an application. The record length is dependent on the number and size of the data fields in the record.

File is a collection of records, not necessarily all the same in length or type. File length is dependent on the number of records in the file. A file may extend over more than one tape reel.

Dictionary is a collection of information which defines the structure of a file by describing each record type and each data field within the file. (The dictionary is used by AIDS to convert external field titles to specific physical locations of data within the file.)

Modes of Operation

AIDS has two modes of operation on the G-635: a time-sharing subsystem and a batch subsystem. Requests or queries are formulated and checked by using the Time-Sharing Subsystem (TSS) of AIDS. This AIDS subsystem allows a user to formulate queries and have the syntax checked via a remote terminal. The queries are grouped together by the system and periodically the batch processing subsystem of AIDS is run to search the data files associated with the stored queries and to produce the required reports. This batch processing subsystem is designed for queries against very large data files, since searching these files on-line would result in extremely long response times; i.e., minutes or hours.

AIDS TSS SUBSYSTEM

The TSS subsystem of AIDS provides the capabilities and facilities for direct query formulation, computer-aided instruction or tutorials for query formulation, and error diagnostics.

System Startup and Termination

The user accesses the AIDS TSS subsystem through the G-635 General Timesharing System (GE-TSS). The AIDS TSS subsystem is defined as a system under GE-TSS. Upon completion of query formulation, the user ends the transaction using normal GE-TSS sign-off procedures. User startup and signoff are outlined in the second subsection of Section IV.

Query Formulation

Queries are submitted on-line via teletype in an English-like language with syntactic validation occurring at the time of entry. All validated queries are stored on disk. File searching and report production are done at a later time by the batch processing subsystem of AIDS.

Available to the user in formulating his queries are facilities for comprehensive data qualifications, sorting, computations, totaling, and subtotaling on data fields within the file. Also available are a reformatting capability for creating a new file subset on tape or cards and a flexible report formatting capability. Additionally, queries used to generate reports may be saved on disk and run again when desired.

System Tutorials

AIDS provides a range of user tutorials for both novice and experienced users. This capability is divided into two basic modes: the syntax validation mode which provides a minimum of user instruction, and the tutorial assistance mode which provides detailed instructions for query development.

Within the syntax validation mode, each line of user input is checked for correctness at the time it is entered. The system then waits for the next line with no intervening instruction. Exceptions to this are error messages and certain phrase boundaries.

The tutorial assistance mode provides three sub-modes of operation. For the completely uninitiated user, the system provides a read-only mode which describes the syntax in detail but does not allow any user query formulation. For slightly more knowledgeable users, the system provides a practice mode which allows queries to be formulated but not run. This mode provides prompting between each line of input which describes the legal syntax elements which can be used at that point in the query. Finally, the system allows for use of the prompting mode for queries which are to be run against data files.

Error Diagnostics

Error diagnostics are available to the user in developing his queries. During all query formulation, input lines may be terminated at any word boundary, at which time a partial syntax check is performed by the system. Error messages describe as completely as possible the remedial actions necessary. On-line query correction is allowed.

BATCH SUBSYSTEM

Major portions of AIDS operate in a batch mode, such as the batch processing of queries and the batch processing utility routines. The utility routines provide functions necessary to build and maintain the internal file used by AIDS while operating under time-sharing. These functions include dictionary definition and maintenance, security file definition and maintenance, and message and tutorial file maintenance.

Batch Processing of Queries

Queries accepted by AIDS and stored on disk are run later in a batch processing mode to retrieve and output data. This batch job is normally run with other queries since the batch subsystem is capable of combining up to ten retrievals in a single pass against a file. The batch nature of the retrieval process makes large file searches more practical since it can be scheduled for periods of low system activity.

Dictionary Definition and Maintenance

The user defines the physical structure of his file to the system in a procedure called Dictionary Definition. Output from the Dictionary Definition Operation, discussed in Section II, is a dictionary file which relates each English title for a data field to the data field's logical and physical position in the file. The user, when naming a data field in his query, utilizes the dictionary file to validate the file and data fields being referenced. The dictionary describes the field layout within a record, allows the various control parameters established by the user to be properly interpreted, and converts codes carried by the tape file into understandable words or abbreviations.

Capabilities exist in AIDS to create, to delete, and/or to modify a dictionary file via card input to the Dictionary Definition Operation. These capabilities are further discussed in the File Modification subsection of Section II and the AIDS Operations Manual.

Security File Definition and Maintenance

AIDS provides the capability to build and maintain a security file for each valid user identification. Each security file contains the file names and dictionary codes associated with that identification.

The security files are typically maintained by each group or agency that has a set of files defined to AIDS. The card inputs to define and maintain each security file are described in the AIDS Operations Manual.

Message File Definition and Maintenance

The message file maintained by AIDS contains two types of messages:

i. Error messages describing user inputs to AIDS.

ii. Tutorial prompting messages aimed at developing user skills in query syntax formulation.

AIDS provides the capability to add, to delete, and/or to modify any or all of the message file statements via card input to the message file processing routine. The message file is further discussed in the AIDS Operations Manual.

SECTION II

FILE DICTIONARY DEFINITION

INTRODUCTION

Each record type format in the data file must be defined to the system. Serial files which consist of fixed-format, fixed-length records can be defined as AIDS data files. The only requirement is that each record type (in multiple record type files) have a unique identifier as one of its data fields. Field titles are assigned to the data fields which the user expects to query. Length, position, coding type, and report field heading are specified for each data field. After the data file definition has been validated by the dictionary utility routine of AIDS and output in the dictionary file, all AIDS query and tutorial functions can operate against the data file.

Logical File Structure

AIDS was designed for use against serially-formatted tape files. The logical structure of files permitted by AIDS can be explained with the use of specific examples concerning a sample file containing personnel and organizational data.

A flat file can be defined to AIDS. A flat file is one which contains one or more record types which are logically independent. For instance, consider a sample file which reports each employee in an organization on a separate record. This file is a single-record type file as illustrated in Figure II-1A. The data field titles are also illustrated. Each record is logically independent; the input order of the records is unimportant to the system.

A flat file may also contain more than one record type, as in the case of a file which contains both employee and organization data. Figure II-1B illustrates the two record types which make up this file as well as the field titles for the data field. Each personnel and organization record is indicated by the codes 101 and 102 in the first data field of each record, respectively. This structure is, in effect, two separate files merged into a single tape file. Again, each record type is logically independent, and the input order of the records is unimportant.

RECORD FORMAT

101	NUM	NAME

SAMPLE DATA RECORDS

101	32579	LYNN, K.R.
101	57060	CARR, P.I.
101	15324	CALLEGAN, R.E.

(A) SINGLE RECORD TYPE, SINGLE LEVEL FILE

RECORD FORMAT

101	NUM	NAME	
102	DIV	TYPE	TITLE

SAMPLE DATA RECORDS

101	32579	LYNN, K.R.	
101	57060	CARR, P.I.	
102	2000	DIV	REPRESENTATIVE DIVISION
101	15324	CALLEGAN, R.E.	
102	2100	DEPT	DEVELOPMENT DEPARTMENT

(B) SINGLE LEVEL, MULTI-RECORD TYPE FILE

IA-34,540

Figure II-1 SAMPLE FILE STRUCTURES

AIDS permits hierarchical-tree or multi-level files of up to eight levels, where each level or node in the tree represents a particular record type. For example, in the sample file a logical structure could be defined which assigns a personnel data record (P) to each personnel record and an organization data record (OD) to each organization record. In structure, the sample appears as illustrated in Figure II-2. More than one P record can exist for each 101 record. Similarly, more than one OD record can exist for each 102 record.

No explicit linking between records of different types or between files is allowed. That is, the system itself cannot be made aware of any relationships between records of different types but relies on the input order of the records to define the hierarchical order. The sample hierarchical file in Figure II-2 is acceptable to AIDS where the 101 and 102 nodes of the tree are first-level records and the P and OD nodes are second-level records. Any meaningful order of the file is reflected in the input order of the records, for the 101 and P records are logically related and the 102 and OD records are, likewise, logically related.

Physical File Structure

AIDS was designed to accept files generated in an external environment. The intent was to accept as many varieties of formats as possible. The system is intended for files generated in a tape-oriented batch processing environment; that is, fixed-record length, serial tape files. Its ability to handle files of mixed-record types of different lengths is particularly useful. It is not intended for and cannot accept complex disk files with indices or hash-code tables.

The file size and other file parameter limitations within AIDS are as follows:

i. The number of records in a file is aribtrarily large since files may extend over more than one physical tape reel. However, the tapes must be recorded in System Standard Format compatible to GECOS. In other terms, the tape may be either a 7- or 9- track tape, but it must be recorded in physical blocks of 320 words (1920 characters).

ii. Each field in the data tape must be character (BCD) coded.

RECORD FORMAT

101	NUM	NAME		
	P	ORG	JOBTITLE	LEVEL	SALARY
102	DIV	TYPE	TITLE	
	OD	REPORTS	COMPOSITION	

SAMPLE DATA RECORDS

101	32579	LYNN, K.R.		
	P	2000	5210	EMPL	6000
101	57060	CARR, P.I.		
	P	2000	1110	HEAD	24000
101	15324	CALLEGAN, R.E.		
	P	2100	5210	EMPL	5400
102	2000	DIV	REPRESENTATIVE DIVISION	
	OD	1000	21002300	
102	2100	DEPT	DEVELOPMENT DEPARTMENT	
	OD	2000	21102120	
	OD	2000	2130	
	OD	2000	2190	

IA-34,541

Figure III-2 SAMPLE HIERARCHICAL FILE STRUCTURE

iii. Maximum size of a data file record is 1,800 characters, with a limitation of 4,968 characters on the sum of all those record lengths at any given file level. As mentioned earlier, there can be a maximum of eight levels in any file. In addition, the system allows for a maximum of 43 record types per file and 999 data fields per level, whose field titles must be unique within the file.

AIDS requires that data files be stored on tape in a serial format. In the case of a hierarchical file, the order of the records on the data tape should be by complete tree-branch top-to-bottom, the order of individual branches being arbitrary. A sensible order for the sample file would be that in Figure II-2. The system requires that for each record type within a file a unique code of one-to-three characters appear in the same position in each record of that type. For instance, the first data field of each record type in Figure II-2 contains a unique code.

The system does not check the input order of records. Consequently, care must be exercised to insure that the file is not in garbled order; otherwise, output will in general be meaningless. Since the files are in serial order on tape, no inversion or randomizing is possible. This implies that access to specific single records whose identification is known in advance requires a search of the entire file.

FILE DEFINITION PROCEDURE

The structure and format of a given file is made known to AIDS via a dictionary. The dictionary describes the data only as it appears within the data file at retrieval time rather than as the data appears for input to a file-generation program.

There are three major sections within an AIDS dictionary: the File Definition Section, the Level Structure Section, and the Field Definition Section. The File Definition Section, consisting of one card per file, defines the file name of the file, the maximum size in words of the largest record in the file, and the number of levels in the file.

The Level Structure Section, consisting of one card per record type, defines the level of the record type involved, the record size, and the position within the record of the record-type code.

The Field Definition Section, consisting of two cards of 29 fields per data field, defines the name of the data field, the position of the data field within the record, its type, and its name for the purpose of report headings.

An optional capability to define coded field lookup tables is provided in a Table Definition Section. The Table Definition Section, consisting of one card per table lookup value, defines a table lookup reference code (or table name), an argument (or data value as it appears in the file), and a function (or value to be used on output). A corresponding table lookup code is added to the data field definition cards if this option is elected.

Coded field lookup tables provide a means of representing the value of a field on output listings in a format different from that used on the data tape. For example, for the field named "SKILL", the values stored in the data file might be: 01, 02, ..., 09, whereas the output values would be: PROGRAMMER, SYS/ANALYST, ..., DEPT. HEAD.

AIDS dictionaries are converted from their external card format to the internal format using the dictionary maintenance program. This batch program generates a new dictionary on tape, adds or deletes fields from an existing dictionary on tape, provides listings of a dictionary for users, or transfers dictionaries from tape to disk for use in processing queries. Dictionaries are normally maintained on tape and reloaded to disk whenever they have been changed externally on tape or accidentally lost on disk.

The following section explains the dictionary card input required for the two-level PERSONNEL file. Formats for the file name, level, and data field card input are provided in each section as well as in Appendix II. Figures II-3, II-4, and II-5 show the sample coding sheets for the PERSONNEL file.

File Name Card

The file name card provides the identification of a particular file to AIDS. Only one card per file is required. The first line of the sample coding sheet for the PERSONNEL file in Figure II-3 provides an example of a file name card. The file name card contains eleven fields as outlined in Table II-1 and is described as follows:

JOB PERSONNEL / AIDS

BY C.A. MARCUS DATE 6/1/71

GENERAL PURPOSE CARD PUNCHING FORM

PAGE 1 OF 3

CHECK ONE
LETTER O SLASHED ✓ NUMBER O SLASHED ___

PUNCHING INSTRUCTIONS

WRITTEN AS:

PUNCH AS:

NOTES:

FIELD IDENTIFICATION

1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
PSNL000001	PERSØNNEL				0001400202	0456 2	
PSNL010101				0001000301	01	0000014	
PSNL010201				0001000301	02	0000014	
PSNL010P 01				0001000011	201	0000014	
PSNL010Ø D 01				0001000202	202	0000014	
PSNL110101	CARD1			0001000301	301	01	
PSNL110121	CARD TYPE				12	0006 0	
PSNL110221	NUM			0007001	1 05	01	
PSNL110221	BADGE				08	0008 0	
PSNL110321	NAME			0013003	7 125	01	
PSNL110321	EMPLØYEE				11	0028 0	
PSNL110401	SEX			0039003	9 101	01	
PSNL110421	SEX				06	0004 0	
PSNL110501	BD			0041004	8 108	01	
PSNL110521	BIRTHDATE				12	0011 0	
PSNL110601	LØC			0050005	4 105	01	
PSNL110621	LØCATIØN				11	0008 0	
PSNL110701	PHØNE			0056006	2 107	01	
PSNL110721	TELEPHØNE				12	0010 0	
PSNL110801	UPD1			0073008	0 108	01	
PSNL110821	UPDATE				10	0011 0	
PSNL209091	ØRG			0007001	0 104	201	
PSNL209221	ØRG				06	0007 0	
PSNL210101	JØBTITLE			0012001	5 104	201	
PSNL210121	JØB TITLE				12	0007 0 0 1 0 2 0	
PSNL211101	LEVEL			0017002	0 104	201	
PSNL211221	LEVEL				08	0007 0	
PSNL212101	SALARY			0022002	7 106	201	
PSNL212221	SALARY				09	2000 0	
PSNL213101	SKILL1			0029003	2 104	201	
PSNL213201	SKILL1				09	1000 7 0 0 1 0 2 0	
PSNL214101	SKILL2			0033003	6 104	201	
PSNL214201	SKILL2				09	1000 7 0 0 1 0 2 0	
PSNL215101	SKILL3			0037004	0 104	201	
PSNL215201	SKILL3				09	1000 7 0 0 1 0 2 0	

MCF 1232

Figure II-3 SAMPLE FILE DEFINITION CARD FORMATS

JOB PERSONNEL / AIDS

BY C.A. MARCUS DATE 6/1/71

GENERAL PURPOSE CARD PUNCHING FORM

PAGE 2 OF 3

CHECK ONE
LETTER O SLASHED ✓ NUMBER O SLASHED ___

PUNCHING INSTRUCTIONS

WRITTEN AS:	
PUNCH AS:	

NOTES:

FIELD IDENTIFICATION

1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
PSNL201601	SKILL4				0041004104201		
PSNL201602	SKILL4				0911000701012O		
PSNL201701	SKILL5				0045004810420 1		
PSNL201702	SKILL5				0911000701012O		
PSNL201801	SKILLS				0029004812020 1		
PSNL201802	SECØNDARY SKILLS				1911002300		
PSNL201901	UPD2				0073008010820 1		
PSNL201902	UPDATE2				1011001100		
PSNL202001	CARD2				0001000110120 1		
PSNL202002	CARD TYPE				0811000400		
PSNL102101	DIVISIØN				0007001010410202		
PSNL102102	DIVISIØN				1111000700		
PSNL102201	TYPE				0011200116105 1 02		
PSNL102202	TYPE				0171000800		
PSNL102301	TITLE				0018004713010202		
PSNL102302	TITLE				0811003300		
PSNL102401	EMPLØYEES				0049005310510202		
PSNL102402	EMPLØYEES				1211000800		
PSNL102501	BUDGET				0055006411010202		
PSNL102502	BUDGET				0921000130		
PSNL102601	UPD3				0073008010810202		
PSNL102602	UPDATE3				1011001100		
PSNL103501	CARD3				0001000310310202		
PSNL103502	CARD TYPE				1211000600		
PSNL202701	CØMPØSITIØN				0011200311202202		
PSNL202702	CØMPØSITIØN				1411002300		
PSNL202801	REPØRTS				0007001010410202		
PSNL202802	REPØRTS TØ				1311000700		
PSNL202901	UPD4				0073008010820202		
PSNL202902	UPDATE4				1011001100		
PSNL203001	CØMP1				0011200115104202		
PSNL203002	CØMP1				0811000700		
PSNL203101	CØMP2				0011600119104202		
PSNL203102	CØMP2				0811000700		
PSNL203201	CØMP3				0012000213104202		
PSNL203202	CØMP3				0811000700		

MCP 1282

Figure II-4 SAMPLE FILE DEFINITION CARD FORMATS

GENERAL PURPOSE CARD PUNCHING FORM

PAGE 3 OF 3

JOB PERSONNEL / AIDS

BY C.A. MARCUS DATE 6/1/71

CHECK ONE
LETTER O SLASHED ✓ NUMBER O SLASHED ____

PUNCHING INSTRUCTIONS
WRITTEN AS:
PUNCH AS:

NOTES:

FIELD IDENTIFICATION

1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
PSNL2033 1 I	COMP4				00240021 I 042		
PSNL2033 2 I	COMP4					081 0001 0	
PSNL2034 1 I	COMP5				00280031 I 042		
PSNL2034 2 I	COMP5					081 0001 0	
PSNL2036 1 I	CARD4				00010012 I 022		
PSNL2036 2 I	CARD TYPE					121 0001 0	
PSNL9010 001 I	0110	ADM. DIV					
PSNL9010 001 I	0130	ADM. DEPT					
PSNL9010 001 I	0150	ADM. GROUP					
PSNL9010 001 I	1110	SYSTEMS ENGR					
PSNL9010 001 I	1120	MECH ENGR					
PSNL9010 001 I	1124	STRUCTURES ENGR					
PSNL9010 001 I	1127	POWER ENGR					
PSNL9010 001 I	1130	ELEC ENGR					
PSNL9010 001 I	1135	COMMUN ENGR					
PSNL9010 001 I	1150	PROD ENGR					
PSNL9010 001 I	1175	METALLURGIST					
PSNL9010 001 I	1180	CHEMIST					
PSNL9010 001 I	1190	SCIENTIST					
PSNL9010 001 I	1330	DRAFTSMAN					
PSNL9010 001 I	1350	CØST ESTIMATØR					
PSNL9010 001 I	1351	MECHN TECHN					
PSNL9010 001 I	1365	TØØL DESIGNER					
PSNL9010 001 I	3125	TØØL MAKER					
PSNL9010 001 I	3320	SCHEDULER					
PSNL9010 001 I	3340	MACHINE ØPR					
PSNL9010 001 I	3345	MILL MACHINE ØPR					
PSNL9010 001 I	3355	PLATING ØPR					
PSNL9010 001 I	3360	HEAT TREAT ØPR					
PSNL9010 001 I	3370	PAINTING ØPR					
PSNL9010 001 I	3510	ASSEMBLER					
PSNL9010 001 I	3590	DISPATCHER					
PSNL9010 001 I	3550	EXPEDITØR					
PSNL9010 001 I	3745	STØCK CLERK					
PSNL9010 001 I	5210	SECRETARY					
PSNL9010 001 I	5520	FILE CLERK					
PSNL9010 001 I	6625	SALESMAN					

MCF 1232

Figure II-5 SAMPLE FILE DEFINITION CARD FORMATS

TABLE II-1

File Name Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	FILLER	5-9	5	Zero Fill
3	CARD ACTION	10	1	I, M, or D
4	FILLER	11-12	2	Blank Fill
5	FILE NAME	13-48	36	Alphanumeric BCD File Name
6	FILLER	49-51	3	Blank Fill
7	MAX RECORD SIZE	52-57	6	Maximum Words in a Record
8	LEVEL	58-60	3	Number of Levels in File
9	FILLER	61-63	3	Blank Fill
10	SEQ KEY	64-69	6	Information Field
11	FILLER	70-80	11	Blank Fill

i. Field 1 defines a four-character alphanumeric code name for the file; for example, PSNL. This code name is repeated as the first field on every card input to the dictionary definition for the file.

ii. Field 2 is always zero-filled and is five characters in length.

iii. Field 3 specifies the action to be taken with the file name card by the dictionary maintenance module. Acceptable one-character codes for this field are:

1. "I" to insert the card to a given dictionary.

2. "M" to modify a previously input file name card.

3. "D" to delete a previously input file name card.

iv. Field 4 is two characters in length and is blank-filled.

v. Field 5 contains an alphanumeric, left-justified file name for the file up to 36 characters in length with no imbedded blanks; for example, PERSONNEL.

vi. Field 6 is three characters in length and is always blank-filled.

vii. Field 7 is a right-justified, six-digit number with leading zeroes which defines the maximum number of words in any record in the file.

viii. Field 8 is a right-justified, three-digit number with leading zeroes which defines the number of levels in the file.

ix. Field 9 is three characters in length and is always blank-filled.

x Field 10 specifies the six-character, alphanumeric sequence key, which may contain any arbitrary information code. This key is printed as part of each dictionary listing heading.

xi. Field 11 is eleven characters in length and is always blank-filled.

Level Structure Card

The level structure card defines each record type in a file according to its level in the file. The user, in defining his file, determines the level structure, if any, within the file. If the file is a single-level, single-record type file, then only one level structure card is required. If more than one level exists in the file structure or if more than one record type exists within a single-level file, one level structure card per record type is required by AIDS. For example, in the sample PERSONNEL file there are two levels and four record types. The first level contains two record types, the personnel record (101) and the organization record (102); and the second level has two record types, the personnel data record (P) and the organization data record (OD), corresponding to the 101 and 102, respectively. The level structure cards for the PERSONNEL dictionary are included in Figure II-3, lines three through six. The level structure card contains 13 fields as outlined in Table II-2 and is described as follows:

i. Field 1 contains the four-character code name of the file exactly as in the first field of the file name card: PSNL.

ii. Field 2 is one character in length and is always zero.

iii. Field 3 specifies the unique one-to-three character record identifier exactly as it appears in the record. The identifier is left-justified in the level structure card and followed by blanks if it is less than three characters in length. If the file being defined to AIDS is a single-level, single-record type file, then this field is filled with three asterisks (***).

iv. Field 4 is one character in length and is always zero.

v. Field 5 is a one-character code which specifies the action to be taken with this level structure card: I, M, or D (insert, modify, or delete).

vi. Field 6 is 38 characters in length and is always blank-filled.

vii. Field 7 contains a four-digit, right-justified number with leading zeroes which defines the starting character position of the record identifier of Field 3.

viii. Field 8 contains a four-digit, right-justified number with leading zeroes which defines the ending character position of the record identifier of Field 3. The number of characters specified by Fields 7 and 8 must match the length of the record identifier.

TABLE II-2

Level Structure Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	FILLER	5	1	Zero Fill
3	RECORD ID	6-8	3	Alphanumeric Code for Record
4	CARD TYPE	9	1	Zero Fill
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-48	38	Blank Fill
7	START LOCATION	49-52	4	Starting Character of Record ID Field
8	END LOCATION	53-56	4	Ending Character of Record ID Field
9	FILLER	57-59	3	Blank Fill
10	DICTIONARY LEVEL CODE	60-62	3	Alphanumeric Dictionary Code for Record
11	FILLER	63	1	Blank Fill
12	RECORD SIZE	64-69	6	Record Size in Words
13	FILLER	70-80	11	Blank Fill

ix. Field 9 is three characters in length and is always blank-filled.

x. Field 10 is a three-character code which defines an internal record type code called the dictionary level code and is made up of two parts. The first digit denotes the level of the record type being defined. The second part is a two-digit code sequentially assigned within each level beginning with the code 01, except in the case of a single-level, single-record type file. In this instance, the first part is "1" and the second part of the dictionary level code must be "**." The dictionary level code must be unique for each record type.

xi. Field 11 is one character in length and is always blank.

xii. Field 12 is a six-digit, right-justified number with leading zeroes which defines the length in six-character (36-bit) words of the record type being defined.

xiii. Field 13 is eleven characters in length and is always blank-filled.

Field Definition Card

The user, in defining data fields, determines a set of data fields within each record type which he wishes to use in generating queries. It is not necessary to define each character in the record. The order of definition of data fields within each record type can be arranged for user convenience and need not reflect the relative order of the data fields within the record. Overlapping data fields may also be defined.

Two cards are required for each field definition. The first card is the field definition card; the second is the report field definition card described in the next subsection of this section. Sample card inputs for both of these cards for the PERSONNEL dictionary are contained in Figures II-3, II-4 and II-5. The field definition card contains 13 fields as outlined in Table II-3 and is described as follows:

TABLE II-3

Field Definition Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	LEVEL	5	1	Record Level
3	FIELD NUMBER	6-8	3	Numeric Identification of Data Field
4	CARD TYPE	9	1	Always "1"
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-12	2	Blank Fill
7	FIELD NAME	13-48	36	Data Field Name
8	START LOCATION	49-52	4	Starting Character of Data Field
9	END LOCATION	53-56	4	Ending Character of Data Field
10	SEARCH TYPE	57	1	I, M, or U
11	FIELD SIZE	58-59	2	Character Length of Data Field
12	DICTIONARY LEVEL CODE	60-62	3	Dictionary Level Code
13	FILLER	63-80	18	Blank Fill

i. Field 1 defines the alphanumeric code name of the file and is the same as in the file name card and level structure card: PSNL.

ii. Field 2 defines the one-digit level of the record in which the data field exists. Up to eight levels are allowed in a file definition.

iii. Field 3 contains a three-digit, right-justified field number with leading zeroes for the data field. AIDS allows a maximum of 999 fields in a level of a file definition. Each field in the definition is assigned a field number unique within a level, and the field numbers need not be consecutive.

iv. Field 4 is always the digit "1" to indicate that the card is a field definition card.

v. Field 5 is a one-character code which specifies the action to be taken with this field definition card: I, M, or D (insert, modify, or delete).

vi. Field 6 is two characters in length and is always blank-filled.

vii. Field 7 defines the data field title. Rules for defining a field title are:

1. Field titles must be unique within the file.

2. A field title may be a maximum of 36 characters in length with no imbedded blanks.

3. Field titles must be entered left-justified in the field followed by blanks.

4. A completely numeric set of characters or an already-existing field title may not be used in combination with any of the critical delimiters. The critical delimiters include the period, colon, semi-colon, left or right parenthesis, relational operators (<, >, =), and computational operators (+, -, /, *). For example, the characters "1-OB" are illegal as a field title, as the digit one precedes a minus or dash sign. However, the characters "OB-T3" are a legal title, as long as "OB" or "T3" are not existing titles in the file.

viii. Field 8 is a four-digit, right-justified number with leading zeroes which defines the beginning character position of the data field in the data record.

ix. Field 9 is a four-digit, right-justified number with leading zeroes which defines the ending character position of the data field in the data record.

x. Field 10 is a one-character code which defines the type of search which will be performed on this data field during retrieval by AIDS. Three search types are allowed by the system: I, M, or U:

1. "I" (Index Search). An indexed data field allows a comparison to be made on the first (left-most) n characters in the field; i.e., a subset of characters beginning with the first character. For example, an indexed comparison made on a data field with the value "ABA" would qualify if the selection criteria were "A", "AB", or "ABA".

2. "M" (Multiple Indexed Search). A multiple indexed field allows repeated comparisons to be made on contiguous subsets of n characters within the data field until the data field is exhausted; i.e., imbedded and mutually exclusive subsets of characters. The number n of characters to be searched is determined by the length of the data value used as a selection criterion in a query. For example, a selection criterion of "BA" would cause the following indexed search on a data field containing the value "ABBA". The first two data characters (AB) would be compared against the criterion "BA". Since the characters in the data field (BA) will be compared against "BA", and a match would occur.

3. "U" (Unindexed Search). An unindexed search looks for a pattern in the data field which is preceded and followed by a field boundary and/or blank characters. For example, a comparison made on the data value "JOHN PAUL JONES " would qualify on any of the selection criteria "JOHN", "PAUL", "JONES", "JOHN PAUL", "PAUL JONES", or "JOHN PAUL JONES". At least one blank character must separate parts of the data field; more than one blank character will be ignored by the search. In practice, an unindexed search would only be used in text processing applications.

xi. Field 11 is a two digit, right-justified number with leading zeroes which defines the number of characters in the data field.

xii. Field 12 contains the dictionary level code of the record type in which the defined data field resides. The three-character code must exactly match the dictionary level code contained in the level structure card of the corresponding record type.

xiii. Field 13 is 18 characters in length and is blank-filled.

Report Field Definition Card

All data fields defined to AIDS must have report field headings defined for output in formatted reports. The report field heading may be the same as the field title or it may be different. In either case, the report field headings must be defined to AIDS.

Examples of report field definition cards for the PERSONNEL file are shown in Figure II-3 and II-4. The report field definition card contains 16 fields as outlined in Table II-4 and is described as follows:

i. Field 1 defines the four-character, alphanumeric code name of the file and is the same as in the file name card, the level structure card, and the field definition card: PSNL.

ii. Field 2 defines the one-digit level of the data field whose report heading is being defined and is the same as the second field of the data field's field definition card.

iii. Field 3 repeats the field number of the data field whose report heading is being defined and is the same as the third field of the data field's field definition card.

iv. Field 4 is always the digit "2" to indicate that the card is a report field definition card.

v. Field 5 is a one-character code which defines the action to be taken with the report field definition card: I, M, or D (insert, modify, or delete).

vi. Field 6 is one character in length and is always blank.

vii. Field 7 contains the report field title. The report field title may be a maximum of 46 characters in length with any number of imbedded blanks or critical delimiters. The report field title is left-justified in the field followed by blanks.

TABLE II-4

Report Field Definition Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	LEVEL	5	1	Record Level
3	FIELD NUMBER	6-8	3	Numeric Identification of Data Field
4	CARD TYPE	9	1	Always "2"
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11	1	Blank Fill
7	REPORT FIELD TITLE	12-57	46	Report Field Title
8	FILLER	58-59	2	Blank Fill
9	TITLE CHARACTERS	60-61	2	Character Length of Report Field Title + 3
10	DATA TYPE	62	1	1, 2, or 3
11	DECIMALS IN	63	1	Number of Input Decimal Places
12	DATA FIELD WIDTH	64-66	3	Character Length of Data Field + 3
13	DECIMALS OUT	67	1	Number of Output Decimal Places
14	TABLE LOOKUP CODE	68-69	2	Lookup Table Identification
15	LOOKUP WIDTH	70-72	3	Character Length of Largest Function + 3
16	FILLER	73-80	8	Blank Fill

viii. Field 8 is two characters in length and is always blank-filled.

ix. Field 9 is a two-digit, right-justified number with leading zeroes which defines the width in characters used in printing field titles on output. Field 9 contains the size or character length of the report field title plus the constant 3 to guarantee the spacing of field titles for output.

x. Field 10 is a one-digit code which defines the data type of the data field. There are three allowable data type codes:

1. Alphanumeric. Data fields that contain all alphabetic values, or numeric values with imbedded decimal points or leading blanks should be declared alphanumeric and assigned the code "1".

2. Numeric. Data fields that contain all numeric values with no leading blanks and no imbedded or assumed decimal places should be declared numeric and assigned the code "2".

3. Decimal. Data fields that contain all numeric values with an assumed decimal point and no leading blanks should be declared decimal and assigned the code "3".

xi. Field 11 is a one-digit number which defines the number of decimal places to the right of the assumed decimal point in a decimal data field. A maximum of nine decimal places may be defined. For alphanumeric and numeric data fields, Field 13 is always zero.

xii. Field 12 is a three-digit, right-justified number with leading zeroes which defines the width in characters used in printing data values on output. Field 12 contains the size or character length of the data field width plus the constant 3 to guarantee spacing of data fields for output.

xiii. Field 13 is a one-digit number which defines the number of decimal places to the right of the decimal point which are to be printed on output for a decimal data field. The number of decimal places for a decimal data value may be truncated or extended on output, depending on user specification. A maximum of nine decimal places may be defined for decimal output. For alphanumeric or numeric data fields, Field 13 is always zero.

xiv. Field 14 is the two-character, alphanumeric code of a lookup table of data values to be used for a given data field for output. The lookup table is discussed in the next section. The table lookup code is optional; if no table lookup is defined for a data field, Field 14 is left blank.

xv. Field 15 is a three-digit, right-justified number with leading zeroes which contains the size of the widest function in the lookup table for a given data field plus the constant 3 to guarantee the spacing of table lookup values for output. If no table lookup is defined for a data field, Field 15 is left blank.

xvi. Field 16 is eight characters in length and is always blank-filled.

Table Lookup Card

A lookup table entry consists of an argument-function pair, in which the argument is a valid data value for a given data field in the file, and the function is the character string which can be printed on output. A lookup table may contain as many entries as there are valid data values for each field using the table. Each entry is defined on a separate table lookup card.

The lookup table is optional. Only one lookup table may be defined per data field, but more than one data field may use a given lookup table. The lookup table is defined by a two-character alphanumeric code which is referenced by those data fields which use the table.

Examples of a complete lookup table definition are shown in Figure II-5. The table lookup card contains 9 fields as outlined in Table II-5 and is described as follows:

i. Field 1 contains the four-character, alphanumeric code name of the file and is the same as in the file name card, the level structure card, and the data and report field definition cards: PSNL.

ii. Field 2 is a one-digit code which defines the card type and is always the number 9.

iii. Field 3 is the two-character, alphanumeric identifier of the table which is the same lookup table code used in the report field definition card. The table lookup code is defined by the user and must appear on each entry of the lookup table.

TABLE II-5

Table Lookup Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	CARD TYPE	5	1	Always "9"
3	TABLE NUMBER	6-7	2	Alphanumeric Identification of Table
4	FILLER	8-9	2	Zero Fill
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-12	2	Blank Fill
7	ARGUMENT	13-24	12	Coded Data Value
8	FUNCTION	25-72	48	Lookup Data Value
9	FILLER	73-80	8	Blank Fill

iv. Field 4 is two characters in length and is always zero-filled.

v. Field 5 is a one-character code which defines the action that is to be taken with the table lookup card: I, M, or D (insert, modify, or delete).

vi. Field 6 is two characters in length and is always blank-filled.

vii. Field 7 contains the coded argument which the data value from the file must match to produce the lookup table function in the report. A maximum of twelve characters may be defined as an argument. The argument is left-justified with trailing blanks.

viii. Field 8 contains the table lookup function or value which will be printed. A maximum of 48 characters may be defined as a function. The function is left-justified with trailing blanks.

ix. Field 9 is eight characters in length and is always blank-filled.

FILE MODIFICATION

Updating an existing file definition by adding or deleting data fields and record types or by changing the characteristics of existing fields is accomplished by card input. Updates to an existing file dictionary are accomplished by inputting the desired changes to the Dictionary Definition Operation outlined in the next section. Input for modification of the file consists of the original file card used to define the file name with the card action field containing the code "M" to indicate that a modification is to be made to the file definition plus any redefinition cards for the dictionary. More than one type of modification may be performed in a single redefinition operation. Redefinitions are input in the order: deletions, additions, modifications.

Add Capability

The user may add a data field and report field, a lookup table or argument-function pair, or a record type to a dictionary by preparing a dictionary card input as outlined in the File Definition Procedure. The following guidelines are outlined to aid proper input when adding to a dictionary:

i. A new file name card cannot be added to an existing file definition. File name cards must always be modified.

ii. The code "I" must appear in the card action field of the card input.

iii. Duplication of field numbers in the field and report field definition cards must be avoided within each level of the file.

iv. Duplication of arguments in the lookup table should be avoided.

Delete Capability

The user may delete a data field and report field, a lookup table or or argument-function pair, or a record type from a dictionary by duplicating and rerunning his dictionary card definitions as originally input to the dictionary with one exception-- a "D" must appear in the card action field of each field. The following guidelines are outlined to aid proper input when using the delete capability:

i. The deletion of a file name card invalidates the entire file dictionary. A more efficient method of deleting a dictionary is to purge that dictionary from the AIDS dictionary file.

ii. If a level structure card (record type) is deleted, all data fields in that record type must be deleted.

iii. If a data field is deleted, the corresponding report field is automatically deleted. Field numbers belonging to deleted data fields need not be redefined, as AIDS does not require consecutive field numbers, just unique ones, within each level of the file.

Modify Capability

The user may modify a data field or report field, a lookup table or argument-function pair, a record type, or a file name by duplicating and rerunning his dictionary card definitions as originally input to the dictionary with two exceptions. First, the card action field must contain an "M" to indicate that the original card definition is being modified. Second, each card input must uniquely identify the original card to be modified. The following guidelines are outlined to aid proper input when using the modify capability:

i. Field numbers plus the file code, the level code, and the card type code serve to uniquely identify the field or report field definition card to be modified. Field number codes may not be modified.

ii. The table lookup code plus the argument, the file code, and the card type code serve to identify the entry definition to be modified.

iii. The record level code plus the file code and the record identifier code serve to identify the level structure definition to be modified.

iv. The combination of file code plus file name uniquely identifies the file name definition to be modified.

DICTIONARY DEFINITION OPERATION

The Dictionary Definition Operation consists of three batch jobs:

i. The dictionary input cards must be validated by the AIDS utility program UPDICT. Output from UPDICT is of two kinds. One is a dictionary listing. Output in the dictionary listings are error messages generated by any invalid input definition cards. Also output is a listing of the valid dictionary containing all fields that are in the dictionary. The second type of output is a tape containing the dictionary.

ii. The dictionary definition output tape from UPDICT must be merged with all other active dictionaries and output to a GECOS permanent sequential file.

iii. The GECOS permanent file must be converted to a random file for AIDS operations on the dictionary.

Further descriptions of the job setups required for the Dictionary Definition Operation are in the AIDS Operations Manual. Included in the descriptions are explanations of the execution decks and GECOS files.

SECTION III

AIDS TSS SUBSYSTEM

INTRODUCTION

The AIDS TSS Subsystem provides capabilities for generating queries against any of the files defined within the system. It accepts and validates the syntax of queries as they are input by the user. System messages are available to guide the user in correcting syntax errors.

QUERY LANGUAGE

The query language, or more precisely, the on-line user language, provides access to all of the functional capabilities of the system, including output formatting and file reformatting. The language is divided into five major functional phrases, each of which corresponds to a major functional capability. These include the following: RETRIEVE, which is the Boolean selection phrase of the language; SORT, which provides the capability for sorting records qualified under the RETRIEVE phrase; COMPUTE, which provides the capability for forming new data elements as an arithmetic function of other elements; PRINT, which provides the capability for output page formatting; and REFORMAT, which allows selected elements of a file to be subset and output to tape or cards.

Syntax diagrams for the major functional phrases are included in the subsection in which each is discussed. The notational conventions used are as follows:

i. Components within braces $\left| \quad \right|$ are required; the user must choose one, and only one, of the components for inclusion in the phrase.

ii. Components within brackets [ ] are optional; the user has the choice of including one, or none, of the components for inclusion in the phrase.

iii. Required phrase words are written in capital letters, and all required words or punctuation are underlined.

iv. Components within parentheses ( ) may be repeated an arbitrary number of times in the phrase.

v. Words, or groups of hyphenated words, with initial capitalization are to be replaced with a user-specified form; the exact forms that may be used are discussed in each section.

vi. Words, or groups of hyphenated words, in lower case lettering require that a valid substitution from the dictionary, namely, a field title or file name, be included in the phrase.

An alphabetic list of the valid AIDS words with the meaning of each word and a page number reference is presented in Appendix I. Optional words are listed with the appropriate phrase description.

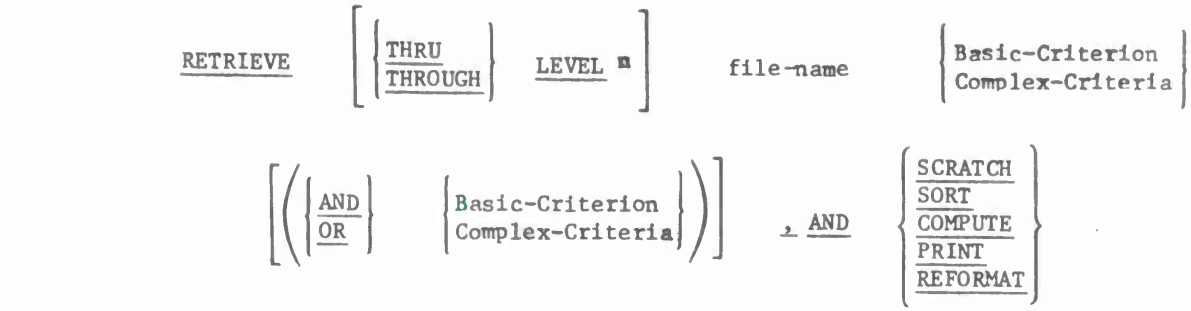
RETRIEVE PHRASE

The RETRIEVE phrase specifies the criteria for selection and retrieval of records from a file. These records are collected on disc by the system and are stored in a file referred to as the hit file. The syntax of the RETRIEVE phrase is diagrammed in Figure III-1. It should be noted that a valid query must include as a minimum a RETRIEVE phrase followed by either a PRINT or a REFORMAT phrase. Valid orders of phrase construction in a query are diagrammed in Appendix IV.

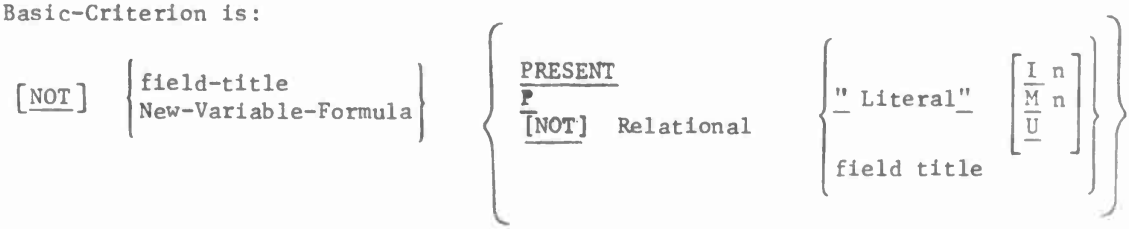
Retrieval Algorithm

The retrieval algorithm used for single-level files (with one or more record types) is straight-forward. Each record in the file is retrieved serially by the system, and selection occurs on each record type that satisfies the selection criteria implied in the qualification portions of the RETRIEVE phrase.

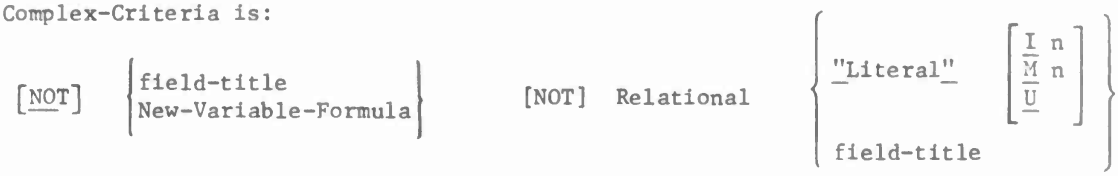
The retrieval algorithm for multi-level files is more complicated. The accurate construction of a RETRIEVE phrase for multi-level file requires that the user understand the logical structure of the file and the logical relationships between records. A multi-level file is searched in groups of records called record sets. A record set is all records which make up a single instance of a complete top-to-bottom branch in a multi-level file. For purposes of illustration, the sample PERSONNEL file from Section II is repeated as Figure III-2. Figure III-2 also shows the field titles assigned to the file's data fields. As in Section II, the sample PERSONNEL file is made up of two levels and four record types. One record set is defined to be an instance of personnel data for each employee; i.e., a first-level 101 type record and an associated second-level P type record. The other record set of this file defined to be an instance of organization data for each organization; i.e., a first-level 102 type record and a second-level OD type

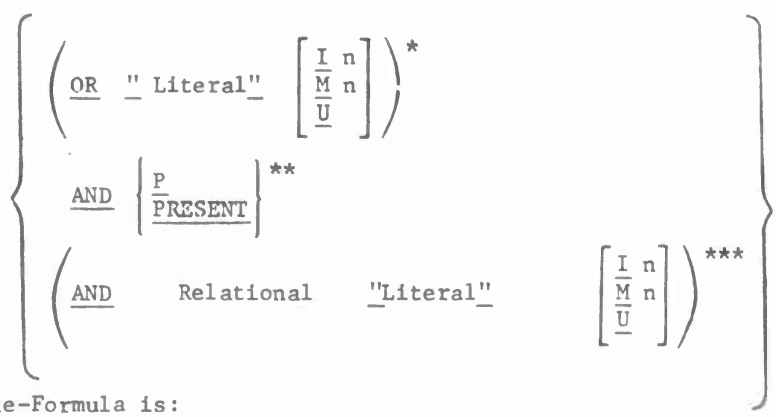


Where Basic-Criterion is:



Where Complex-Criteria is:





Where New-Variable-Formula is:

$New-Variable-Name = Formula

*This syntax form may only be used with an unnegated equals relational.

**This syntax form may only be used with a less than, or less than or equal, relational or its equivalent.

***The syntax form for a range criteria must be both non-repetitive and non-negated.

Figure III-1. RETRIEVE Phrase Syntax

RECORD FORMAT

101	NUM	NAME			
	P	ORG	JOBTITLE	LEVEL	SALARY
102	DIV	TYPE	TITLE		
	OD	REPORTS	COMPOSITION		

SAMPLE DATA RECORDS

101	32579	LYNN, K.R.			
	P	2000	5210	EMPL	6000
101	57060	CARR, P.I.			
	P	2000	1110	HEAD	24000
101	15324	CALLEGAN, R.E.			
	P	2100	5210	EMPL	5400
102	2000	DIV	REPRESENTATIVE DIVISION		
	OD	1000	21002300		
102	2100	DEPT	DEVELOPMENT DEPARTMENT		
	OD	2000	21102120		
	OD	2000	2130		
	OD	2000	2190		

IA-34,541

Figure III-2 SAMPLE HIERARCHICAL FILE STRUCTURE

record. During retrieval from the PERSONNEL file, each complete record set is assembled by AIDS in core memory, and selection of the record set occurs when it satisfies the qualification criteria of the query.

It is usually the case in a two-level file that there are multiple instances of second-level records for each first-level record. For example, in the PERSONNEL file, it is likely that a given organization will have a number of OD records associated with it. Additional OD records for that organization would then be included in the file in the manner illustrated in Figure III-2, with OD records belonging to a single organization record stored contiguously. Processing of successive organization record sets proceeds as follows: AIDS retrieves a 102 record and the first associated OD record as an initial record set. If qualification occurs on this record set, then the whole record set (both records) would be written in the hit file. The next record set is constructed by reading the following record in the serial file, which in the example is another OD record. The second record set then consists of the first 102 record and its second related OD record. If qualification occurs on this second set, it would also be written in the hit file; if not, the record set is overwritten with blanks in core memory. Each successive record set is constructed by retaining the initial 102 record and reading the next record in the file until the set of related OD records is exhausted. Upon encountering a new organization (102 record), AIDS discards the previous 102 record and begins constructing a record set for the new organization. The user will note that a record set may consist of just a 102 record if no OD records exist for that organization. It is clear that a user must understand the structure of his file in order to successfully qualify record sets which he wishes to retrieve. From this example, it is evident that this retrieval algorithm extends to files of more than two levels.

Search Methods

AIDS provides two methods for restricting the normal search of a file. These are the restricted level search feature and the exception reporting search feature.

Restricted Level Search Feature

The normal search method in AIDS examines complete record sets. The restricted level search eliminates all records at a logical file level lower than that specified. The restricted level feature can save processing time in searching a complex file. For example, by restructuring the search to a single level in the PERSONNEL file, any OD or P type records would be excluded from the selection process, as well as from the hit file.

Exception Reporting Search Feature

The exception reporting search feature, also called bracket reporting, allows the user to select all logically related record sets based on the qualification of only one of them. For example, qualification of one occurrence of the COMPOSITION field equal to "2130" in the sample PERSONNEL file would cause all logically related sets to qualify; namely, the record sets with the OD records containing the COMPOSITION fields equal to "21102120", "2130", and "2190". The exception reporting search constructs and saves all record sets that are logically related in a temporary file. If one or more of the record sets qualify, the temporary file is written to the hit file. If no occurrences qualify, the temporary file is scratched.

The exception reporting search feature is invoked in the qualification phrase of the query by enclosing the qualification criteria in brackets [ ]; hence, the name bracket reporting. A capability also exists for performing cross-branch selection by connecting bracketed qualification criteria with the Boolean operators AND and OR.

File Name Specification

RETRIEVE must be the first word of every search phrase. The name of the file to be queried follows the word RETRIEVE. To retrieve records from the PERSONNEL file, the correct input is:

```
RETRIEVE PERSONNEL
```

The file name identifies the file from which hit record sets will be selected and from which output will be generated.

Syntax of Restricted Level Search

To specify a restricted level search within a file, the words THROUGH LEVEL n or THRU LEVEL n, where n is a number from one to eight, must appear immediately after the word RETRIEVE and immediately before the file name. n may be either a digit or the number spelled out:

```
RETRIEVE THRU LEVEL ONE PERSONNEL
RETRIEVE THROUGH LEVEL 1 PERSONNEL
```

In this example, only records in the first level of the PERSONNEL file; i.e., the personnel 101 type record and the organization 102 type record, will be searched.

New-Variable Definition

A temporary new variable can be defined and used in place of field title in the RETRIEVE phrase. A new variable is composed of a user-defined new variable name, an equals relational, and an arithmetic formula. For example:

```
$MO-SALARY = SALARY/12
```

The new variable name establishes the title by which the result of the computation is identified. A dollar sign ($) is required as the first character of the new variable name followed by a string of alphanumeric characters up to 35 in length which does not duplicate any of the file's existing field titles. An all-numeric set of characters may not be used as a new variable name. There must be one character in addition to the dollar sign, and imbedded blanks are not allowed. Hyphens (-) may be used to separate parts of the new variable name.

The new variable name must be separated from the formula by an equals sign (=). The formula is any well-formed arithmetic expression made up of valid field titles, constants, arithmetic operators, and parentheses. Constants are any decimal number, integer, or fraction. Functions may not be referenced within the formula; e.g., SIN (X). The set of valid arithmetic operators are: addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (**). Up to three levels of parentheses may be used.

Orders of Operations

In an expression without parentheses, the calculation is performed with the following precedence for operations:

i. Exponentiation.

ii. Division and Multiplication.

iii. Addition and Subtraction.

The priority of multiplication and division depends on the input order. In the preceding example, the expression .05 * SALARY/12 would be interpreted as .05 multiplied by SALARY, then divided by 12. This same rule of input order is true for addition and subtraction.

Parentheses in an Arithmetic Formula

Parentheses serve to clearly delineate the precedence of operations to be performed in more complex computations. The following example shows this use of parentheses:

```
$MONTHLY-SALARY = SALARY/12 + .05 * SALARY/12

$MONTHLY-SALARY = (SALARY + (.05 * SALARY))/12
```

which reduces the number of operations from four to three. Operations within parentheses are performed first in any function or formula, with the order of processing within each parentheses following the same rules as for that of no parentheses. The order of processing is from the the deepest level of parentheses to the outward level of parentheses.

Basic Criterion

The file name must be followed by at least one basic criterion which specifies which record sets will be inserted in the hit file. A basic criterion is a triplet composed of a field title or new variable, a relational, and a criterion, in that order. The field title is defined in the file dictionary. The relational may be any one of those listed in the RETRIEVE word list in Appendix I. The criterion is a literal or data field against which the data field is to be compared.

Forms of Qualification

Three basic forms of triplets can be input to the system:

i. Field-to-field comparison.

ii. Field-to-literal comparison.

iii. Field present comparison.

The format of each is discussed separately below.

Field-to-Field Comparison. This first form of qualification compares the data values of two fields:

```
RETRIEVE PERSONNEL NUM = BD
```

The value of the data field NUM is compared to the value of the data field BD in each record set containing the fields NUM and BD.

Field-to-Literal Comparison. The field-to-literal qualification compares a literal value (alphabetic, numeric, or alphanumeric) to the value of a data field:

```
RETRIEVE PERSONNEL ORG = "2100"
```

The literal against which the data field ORG is to be compared is enclosed in quotes and must not exceed 48 characters in length. All characters in the literal, including the quotation marks, must be input on one line in AIDS.

Field Present Comparison. The third form of qualification is the field present comparison in which a data field is tested to see if it contains non-blank characters. The statements:

```
RETRIEVE PERSONNEL ORG P
RETRIEVE PERSONNEL ORG PRESENT
```

both qualify those record sets in which the value of ORG is not all blanks.

Field Search Methods

The data field which is to be compared to the criterion can be searched using any of the following methods:

i. Indexed search.

ii. Multiple indexed search.

iii. Unindexed search.

The use of a search criterion in a query overrides the search type defined for a given data field in the dictionary. If no search criterion is specified by the user in his query, the system assumes the search type in the dictionary.

Indexed Search. In an indexed search on a data field, only the left-most characters of the data field are used for qualification. Syntax for an indexed search is of the form I nn, where nn is a number from 1 to 48 establishing the number of characters in the data field which are to be compared to the criterion. Indexed search specification is positioned after the RETRIEVE basic criterion. In the example:

```
RETRIEVE PERSONNEL ORG = "21" I 02
```

the specification I 02 would compare the first two characters in the ORG data field with the criterion "21" to determine if the record set qualifies.

Multiple Indexed Search. A multiple indexed search is identified by an M nn positioned after the RETRIEVE basic criterion where nn is a number from 1 to 48. The data field will be examined from the left starting with the first character, as in an indexed search. However, if there is no qualification on the first nn characters, the next nn characters, starting at character position nn + 1, are tested, and this is continued until the field is exhausted.

Unindexed Search. An unindexed search is identified by a U positioned after the RETRIEVE basic criterion. It is normally used to examine a textual data field for the presence of a word. Unindexed search examines between spaces or blanks within the data field and must find the entire word with blanks or data field boundaries on either side of it, in order for the record set to qualify. For example,

```
RETRIEVE PERSONNEL NAME = "BENJAMIN" U
```

requires that the literal "BENJAMIN", preceded and followed by blanks or data field boundaries, be present in the data field. "BENJAMIN" followed or preceded by a period or comma in the data field would not qualify. If a literal is composed of two words which are separated by an arbitrary number of blanks, the search will look for the two words in the sequence given with at least one intervening blank. More than one blank may separate the words, but these additional blanks are ignored by the unindexed search.

Complex Criteria

Up to 100 triplets may be used in each RETRIEVE phrase. Each triplet in the RETRIEVE phrase must be connected to the succeeding one with AND or OR. It is sufficient to mention here that a basic triplet has a true or false value associated with it. The connectors AND and OR are used as in any Boolean expression and have the same true-false connotations.

Use of Field-to-Field Comparison

Boolean connectors may not be used to connect field titles directly. For example, the phrases:

```
RETRIEVE PERSONNEL ORG AND NUM EQ BD

RETRIEVE PERSONNEL BD = ORG OR NUM
```

are invalid, as ORG and NUM are joined directly by Boolean connectors.

Use of Field-to-Literal Comparison

If the contents of a particular data field are to be compared to each of several values, then the field title may be, but does not have to be, repeated for each comparison. For example, the phrases:

RETREIVE PERSONNEL ORG = "2100" OR ORG = "2000"

RETRIEVE PERSONNEL ORG = "2100" OR "2000"

are identical in meaning, but the second phrase contains the field title ORG only once. OR conditions connect each literal.

Boolean connectors may also be used to link relationals and literals for a given field title in a range comparison:

RETRIEVE PERSONNEL ORG GE "2100" AND LT "2300"

Use of Field Present Comparison

The use of the field present comparison is important when the user wishes to search a multi-level or a single-level, multi-record file using the LT, LESS THAN, LE, or < relationals. In the example:

RETRIEVE PERSONNEL ORG LE "2353"

AIDS would retrieve each record set sequentially in the file. If the record types in the record set are not those implied by the use of the data field ORG, the system will blank the entire record set it has retrieved in core and will continue with the comparison of the data field ORG against the blank record set available. ORG, in this example, would qualify on the blank fields in the record set due to the position of the blank character in the collating sequence of the G-635. This blank record set would then be added to the hit file as a qualifying record set.

To avoid all blank record sets qualifying in the hit file, the following general form should be used with a less than or less than or equal to relational:

RETRIEVE field-title { LESS THAN / LT / LE } "Literal" AND { P / PRESENT }

The criterion PRESENT or P requires that a non-blank data value be in the ORG data field to qualify for selection.

Use of Linked Relationals

Boolean connectors may be used to link relationals. For example, the phrase:

```
RETRIEVE PERSONNEL ORG LT OR = "2100"
```

links the relationals LT and = into a less than or equal to condition, which could be more easily expressed by the relational LE. Generally, the linking of relationals does not add more power to the RETRIEVE phrase, as valid relationals exist which are already combinations of two relationals; e.g., LE and GE. An obvious exception is the PRESENT condition. The statement:

```
RETRIEVE PERSONNEL ORG LT "2100" AND P
```

is an example of the only case where relationals may be linked by an AND condition. All other linked relationals must be joined by the Boolean connector OR.

Parentheses in Boolean Expressions

Parentheses may be used to resolve the ambiguities common to logical relationships. Parentheses within the RETRIEVE syntax phrase must be balanced; that is, each left parenthesis must have an associated right parenthesis. The system allows up to three levels of parentheses. Parentheses must not be placed within complete basic or complex criteria. For instance, in the following example, the parentheses serve to clarify the components of the RETRIEVE phrase; in the first case, a basic criterion, and in the second case, a complex criteria:

```
RETRIEVE PERSONNEL (ORG GT "2122") AND (NUM LT "5100" AND P)
```

Parentheses can serve to resolve logical ambiguities in the search criteria, as well as to decrease the number of components in the search criteria. For example, to query on a given JOBTITLE, and either a SALARY over a certain amount or the ORG to which the JOBTITLE contributes, the following two RETRIEVE phrases are equivalent in meaning:

```
RETRIEVE PERSONNEL JOBTITLE = "2124" AND SALARY GE "10000" OR
JOBTITLE = "2124" AND ORG = "2100" OR "2122" OR "2123"

RETRIEVE PERSONNEL JOBTITLE = "2124" AND (SALARY GE "10000" OR
ORG = "2100" OR "2122" OR "2123")
```

The second RETRIEVE phrase decreases the number of search criteria from six to five.

Negation

The negator NOT may be used to negate any basic search criterion or complex criteria in the RETRIEVE phrase. The word NOT implies the negation of whatever follows in the RETRIEVE phrase and may precede a data field title, a relational, a left parenthesis, and a left bracket. This negation is carried through the phrase until the occurrence of the words AND or OR in a non-parenthesized search phrase, the presence of a right parenthesis in a negated, parenthesized search criteria, or the presence of a right bracket in a negated, bracketed search criteria. The following phrases illustrate the effect of the negator:

```
RETRIEVE PERSONNEL NOT (ORG GT "2100") AND P
RETRIEVE PERSONNEL NOT ORG GT "2100" AND P
RETRIEVE PERSONNEL ORG NOT GT "2100" AND P
```

The phrases are equivalent in meaning; each negates a GT condition into a LE condition.

When NOT is used in a RETRIEVE phrase, the following limitations apply:

i. NOT may not be immediately followed by NOT.

ii. The sequence "(NOT(" may not appear anywhere in the RETRIEVE phrase.

iii. NOT may not immediately precede the connectors AND or OR.

iv. NOT may not immediately precede the words P or PRESENT.

Syntax of Exception Reporting Search

There are two forms of exception reporting: one for cross-branch reporting, and another for single branch reporting. To use the single branch reporting technique, brackets must precede and follow the complex criteria in the RETRIEVE phrase. For example, the phrase:

```
RETRIEVE PERSONNEL [COMPOSITION = "2130"]
```

implies that a single branch exception reporting search is to be done on the file. Note that all the elements in the phrase are enclosed in one set of brackets. Imbedded brackets are not allowed. Parentheses are valid within the brackets, but may not be used external to brackets.

The cross-branch reporting technique is a search technique that will not be explained in this manual. It is sufficient to mention that a cross-branch exception reporting search uses more than one set of brackets, with the search criteria from each branch entirely contained in one set of brackets, and multiple sets of brackets joined by the Boolean connectors AND or OR.

Termination of RETRIEVE Phrase

The RETRIEVE phrase is terminated by a comma followed by the connector AND. The RETRIEVE phrase may be followed by a SORT, COMPUTE, PRINT, REFORMAT, or SCRATCH phrase.

SORT PHRASE

The SORT phrase allows the hit file to be ordered on one or more data fields within it. The SORT phrase is optional and may follow either a RETRIEVE or a COMPUTE phrase (discussed in the next section). The syntax of the SORT phrase is diagrammed in Figure III-3. The notation used is the same as that used in the previous diagram.

Sort Specification

SORT must be the first word of the phrase and must be followed by at least one sort element. A sort element consists of a field title or new variable name, the portion of the sorted data field to be used as the sort key, and the direction of the sort, either ascending or descending.

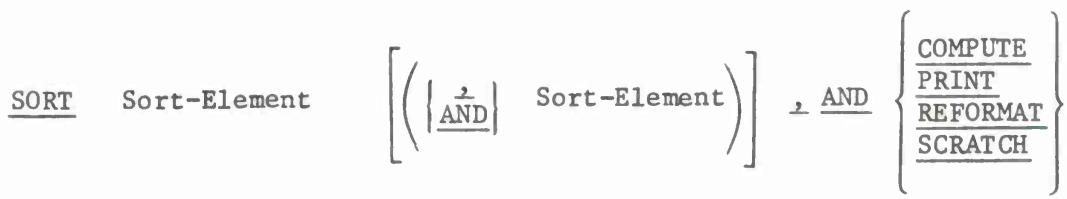
The only mandatory component of a sort element is the field title or new variable name (previously defined in a COMPUTE phrase) to be sorted:

```
SORT ORG
```

The direction of the sort may be specified by following the field title or new variable name by the words ASC, ASCENDING, DSC, or DESCENDING. If no sort direction is specified, an ascending sort is assumed by the system. That portion of the data field to be used as a sort key may be defined by inputting a number n from 1 to 60 preceding the field title:

```
SORT 2 ORG DSC
```

The sort key is defined to be the left-most n characters on which the sort is ordered. The entire data field is used as the sort key if no sub-field is specified.



Where Sort-Element is:

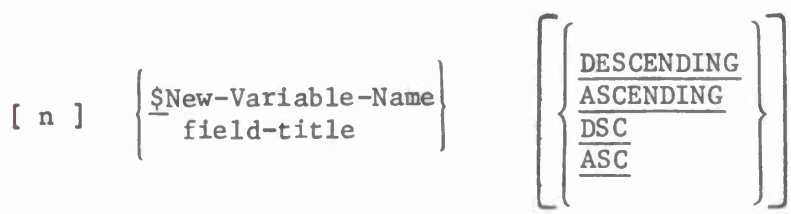


Figure III-3. SORT Phrase Syntax

Multiple Sorts

More than one data field or new variable may be defined as sort keys. Each sort element is separated by commas or by the word AND. When more than one sort element is specified, the first element entered is the major key, and successive keys are ranked according to their input order. For example, the SORT phrases:

SORT ORG DSC, JOBTITLE ASC, 10 NAME

SORT ORG DESCENDING AND JOBTITLE ASCENDING AND 10 NAME

will perform the initial descending sort on the data field ORG, the second ascending sort within each ORG on JOBTITLE, and the final ascending sort within each JOBTITLE on the first ten characters of each name.

Sort Limitations

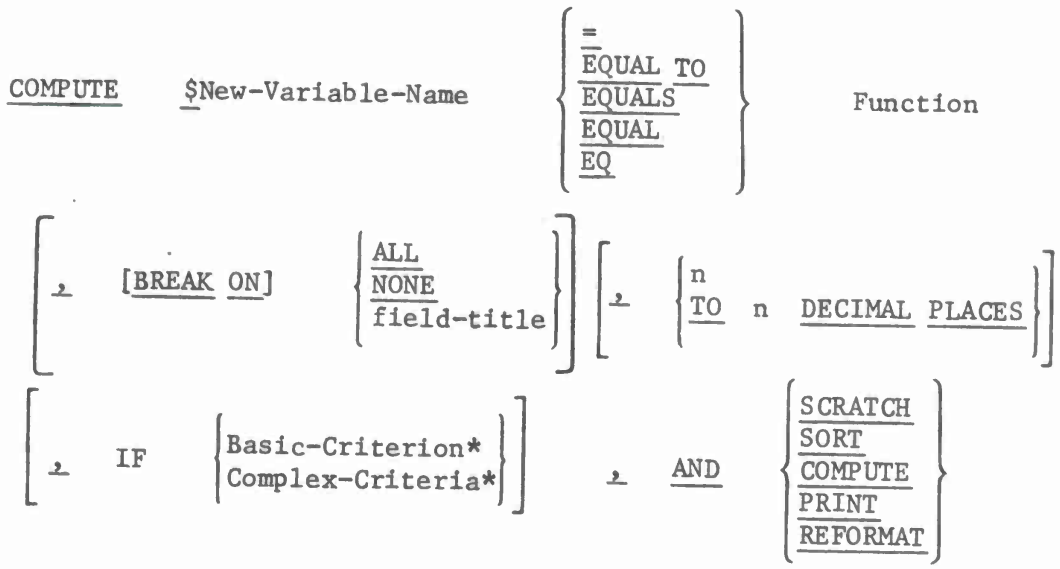
The total number of characters used as sort keys in all elements may not exceed 60 characters per query. The characters may be distributed among the sort elements in any proportion. The capability of sorting on a sub-field provides the user with the ability to sort on data fields whose combined character length is more than the 60 character limitation.

Termination of SORT Phrase

The SORT phrase must be terminated with a comma followed by the connector AND. The SORT phrase may be followed by any of the phrases SCRATCH, COMPUTE, PRINT, or REFORMAT. The user should note that a SORT may be performed before or after a COMPUTE phrase, but the combination of a SORT followed by a COMPUTE followed by another SORT is allowed only once in each query. Also, a SORT phrase may not separate COMPUTE phrases.

COMPUTE PHRASE

The COMPUTE phrase allows the user to define new data fields as arithmetic functions of other data fields and literal values. Precedence of arithmetic operations is the same as in the RETRIEVE phrase. The format of the computed new variable is the same as that described in the RETRIEVE phrase except that any of the valid equivalents for the equals relational may be used (=, EQ, EQUAL, EQUALS, EQUAL TO). The syntax of the COMPUTE phrase is illustrated in Figure III-4. The notation used is the same as that used in earlier diagrams.



*No New-Variable-Formula can be defined in an IF clause.

Figure III-4. COMPUTE Phrase Syntax

SUM and COUNT Functions

In addition to arithmetic formulas, the COMPUTE phrase allows SUM and COUNT functions which have the form:

$New-Variable-Name { EQ | EQUAL | EQUALS | EQUAL TO | = } { SUM | COUNT } field-title

Examples of the use of COUNT and SUM functions are:

```
$SUM-SALARY = SUM SALARY

$NO-EMPLOYEES = COUNT NAME
```

where SALARY and NAME are field titles in the PERSONNEL file. The values of these functions are the sum of field values or a count of non-blank field instances.

The SUM function generates a running total of the data values within a particular data field. This total can be broken or reinitialized when the value of a specified data field changes by using the BREAK option described next. After a break, the sum is reinitialized to zero and SUM continues until the next break occurs. The COUNT function counts the number of occurrences of a given data field. The BREAK option, when used with the COUNT function, establishes when the count is reinitialized to zero.

BREAK Option

Use of the break option in a COMPUTE phrase is designated by the word ALL, the word NONE, or a field title. The "break" defines when a new variable is to be inserted in the hit file and then reinitialized to zero. The word ALL causes the designated computation to be performed for each record set. The word NONE causes no reinitialization to occur during all record set processing. Use of a data field title in a break sub-phrase causes the computation to be performed when the contents of the specified field change in the hit file. If no break field is specified, ALL is assumed by the system. The examples below serve to illustrate each of the break options allowed:

```
COMPUTE $MONTHLY-SALARY = (SALARY + (.05 * SALARY))/12,ALL

COMPUTE $MONTHLY-SALARY = (SALARY + (.05 * SALARY))/12
```

```
COMPUTE $SALARY-BY-ORG = SUM SALARY, ORG

COMPUTE $NO-EMPLOYEES = COUNT NAME, NONE
```

The break phrase is separated from the computation by a comma. The words "BREAK ON", termed non-essential words, may be used to clarify the break phrase if desired:

```
COMPUTE $SALARY-BY-ORG = SUM SALARY, BREAK ON ORG
```

Decimal Significance Option

The decimal significance option is used to establish the number of significant places carried in the computation and printed on output. This is expressed as a single digit numeric 0 to 9 or the words ZERO, ONE, ..., NINE following the formula or function in the COMPUTE phrase and set off from the rest of the phrase by a comma. When the decimal significance is not specified, zero places are assumed; i.e., integral values are calculated. To illustrate:

```
COMPUTE $MONTHLY-SALARY = (SALARY + (.05 * SALARY)) /12, ALL, TWO
```

The above statement specifies that $MONTHLY-SALARY is to be computed to two decimal places for each record set. Each computed new variable is limited to twelve (12) digits in length by the system.

To clarify the decimal significance option, optional words such as TO n DECIMAL PLACES, where n is the number of decimal places, are allowed:

```
COMPUTE $MONTHLY-SALARY = (SALARY + (.05 * SALARY)) /12, TO
2 DECIMAL PLACES, BREAK ON ALL
```

IF Clause Option

The domain of the COMPUTE phrase within the hit file can be modified by using an IF clause. The IF clause option causes calculations to be performed only on record sets meeting the criteria defined within the clause. The word IF must begin the clause followed by any string of complex criteria connected with Boolean ANDs and ORs and containing parentheses (see the RETRIEVE section). For instance, the statement:

```
COMPUTE $MONTHLY-SALARY = (SALARY + (.05 * SALARY)) /12,2,
IF SALARY > "10000"
```

would cause the computation to be performed only on record sets with the value of the data field SALARY greater than 10000. Two decimal places of significance are specified, and a break field of ALL is assumed. The order of input of the COMPUTE options is arbitrary.

COMPUTE Phrase Limitations

Up to 60 COMPUTE phrases may be defined in each query. Each COMPUTE phrase is limited to:

i. 40 components, with new variable names, equals relationals, field titles, parentheses, literals, and arithmetic operators counting as one component each.

ii. 10 constant (literal) values.

iii. 14 field titles.

Multiple COMPUTE Phrases

The user may elect to include more than one COMPUTE phrase in his query. To do so, the user must follow each complete phrase with the connecting sequence ", AND" and the next COMPUTE phrase. The order of the phrases is unimportant. For instance, the phrases:

```
COMPUTE $NO-EMPLOYEES = COUNT NUM, NONE, AND

COMPUTE $AVERAGE = SALARY/12, TO 2 DECIMAL PLACES, AND

COMPUTE $EMP-IN-ORG = COUNT NUM, BREAK ON ORG, IF ORG "20000"
```

define three new variables.

This method of entering multiple COMPUTE phrases provides the user with an additional capability, namely, a previously defined new variable can be used as a field title in any subsequent formula. It cannot, however, be used in a BREAK or an IF clause or as an argument in a SUM or COUNT function. For example,

```
COMPUTE $AVERAGE = SALARY/12, TO 2 DECIMAL PLACES, AND

COMPUTE $MONTHLY-SALARY = $AVERAGE * (.05), TO TWO DECIMAL PLACES
```

the new variable $AVERAGE is used as a data field in the second formula.

Termination of COMPUTE Phrase

The last COMPUTE phrase of the query must be terminated with a comma followed by the connector AND, whereupon the user may proceed to a SORT, PRINT, or REFORMAT phrase, or he may SCRATCH the entire query.

PRINT PHRASE

The PRINT phrase is used to output a formatted hard copy containing values and headings from the hit file. Features included are system or user position specification for headings and data values, full and summary reports, pagination control, table lookups for coded data values, and suppression of repetitive data values. More than one PRINT phrase may be used in each query; each PRINT phrase generates a separate output report. The syntax of the PRINT phrase is diagrammed in Figure III-5. The notation used is the same as that used in earlier diagrams.

System Default Positioning

In its simplest form, the PRINT phrase consists of a list of field titles and new variables separated by commas. When this form of the PRINT command is used, default formatting is in effect, and the output format is tabular with the report field titles across the top of the page and with data values directly beneath them. Spacing between the report field headings is automatically generated as well as spacing between lines of data. Two lines are skipped between the report field heading and the first line of data. One line is skipped between each line of output, and each line of output is positioned at the left-most line boundary. For example, the PRINT phrase:

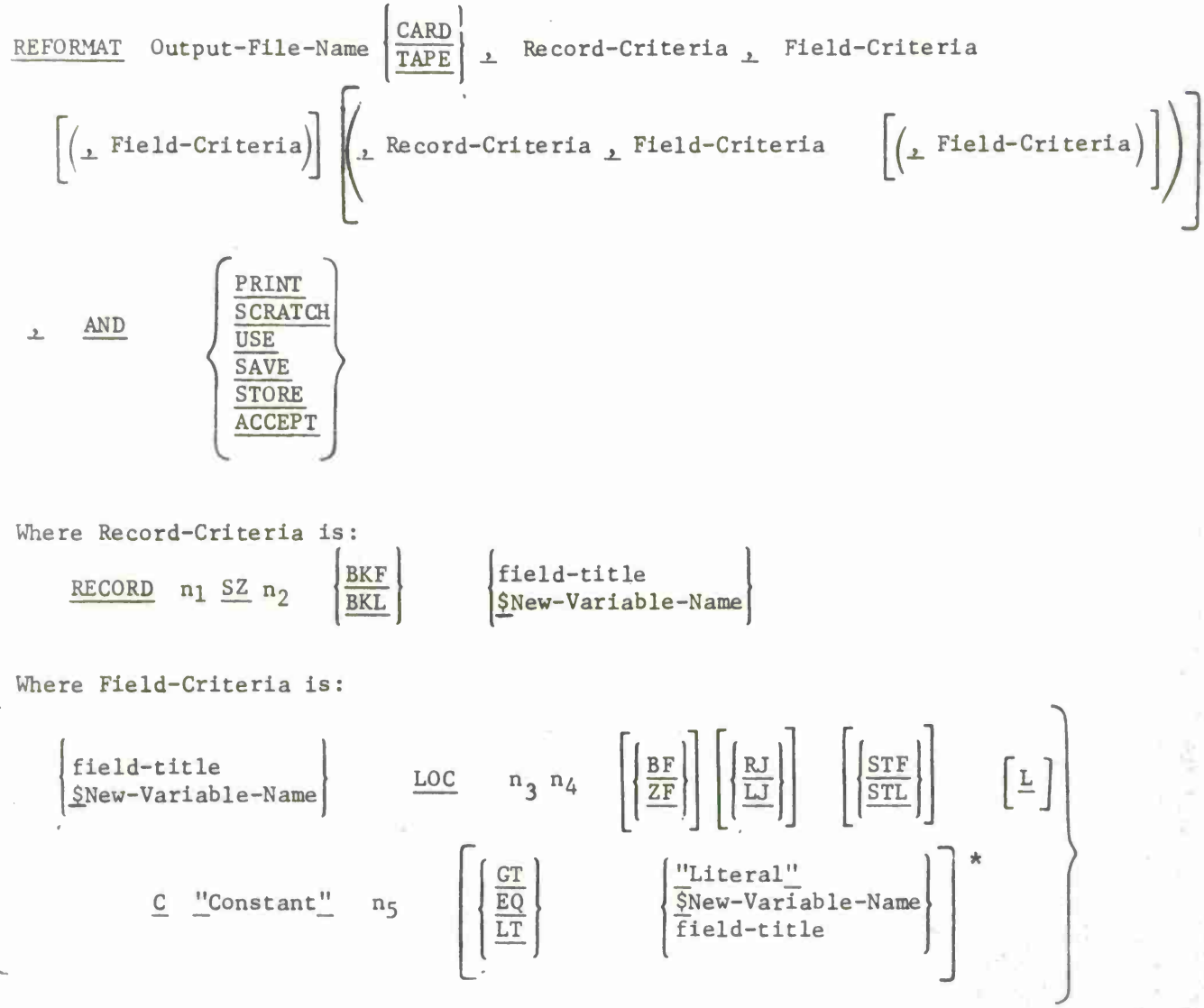
```
PRINT ORG, NAME, JOBTITLE, $MONTHLY-SALARY
```

would yield this output format from the sample PERSONNEL file:

ORG	EMPLOYEE	JOB TITLE	MONTHLY SALARY
2000	LYNN, K. R.	5210	500
2100	CARR, P. I.	1110	2000
2100	CALLEGAN, R. E.	5210	450

User Defined Positioning

Users may override the default formatting of the system to generate formats more suited to their needs. Options available to the user are horizontal positioning, vertical positioning, page ejections, and up to five lines of output per record set based on a 120-character print line.



*At least one output field must be defined before a constant insertion can be defined.

Figure III-6. REFORMAT Phrase Syntax

Horizontal Positioning

The horizontal position of each field on the page is determined by specifying the starting character position (print column) after the field title. Valid subphrases for horizontal positioning within a line are POS n, COLUMN POSITION n, and CHARACTER POSITION n, where n is a number from 2 to 120. A valid PRINT phrase with horizontal positioning specified is:

```
PRINT NUM POS 2, NAME POS 15, SALARY POS 55
```

The user must allow for the maximum number of characters in each data value plus the constant three between each assigned position; a data field may not overlap line boundaries.

Vertical Positioning

Vertical line spacing up to a maximum of eight spaces may be specified between successive print lines. Valid subphrases for vertical positioning within a page are SPACE n, SLEW n, and SKIP n, where n is a number from 1 to 8. Alternatively, a semicolon may be used for spacing to the next line instead of SPACE 1. The new output line, in any case, is positioned from the left-most line boundary unless horizontal positioning has been specified. Vertical positioning subphrases may be included in the PRINT phrase either before or after a field title. Vertical line spacing subphrases appearing before (after) the field title cause line spacing in the output report to occur before (after) the printing of the corresponding data values.

Page Ejection

The inclusion of a vertical spacing subphrase, SLEW 9, SKIP 9, SPACE 9, or TOP before (after) a data field causes a skip to a new page before (after) the next data field is output. For example, the PRINT phrase:

```
PRINT TOP NAME, NUM, SALARY
```

causes each group of 3 values of output to be printed on a new page.

Multiple Output Lines

Output from each record set is limited to a total of five printed lines, each a maximum of 120 characters in length. This implies a maximum of 600 printed characters per record set, but spacing between data fields typically limits this to approximately 500 characters.

To produce multiple lines of output from a record set using default positioning, the user inputs his list of data field titles in the PRINT phrase and relies on the system to position the output within each 120-character line. Report field headings will reflect the levels of output automatically. Default positioning does not split data fields over line boundaries.

User specified positioning allows more flexibility for structuring lines of output from a record set. The user can generate multiple lines of output by defining vertical line spacing either before or after a data field title, and by positioning subsequent data fields within the new line. For instance, the statement:

```
PRINT NUM; SALARY POS 30; NAME POS 50
```

will position each occurrence of NUM, SALARY, and NAME as follows:

```
BADGE
                SALARY
                                EMPLOYEE
32579
                 6000
                                LYNN, K. R.
57060
                24000
                                CARR, P. I.
15324
                 5400
                                CALLEGAN, R. E.
```

The occurrences of NAME, NUM, and SALARY within a record set are positioned on separate lines. The data values are centered beneath the field titles as shown.

Output Devices

The user should be aware of the line length restriction of each type of output device when formatting a report. The line printer allows up to five lines of output per record set, where each line is a maximum of 120 characters in length. The teletype also allows 600 characters per record set for output where the output is distributed over 10 lines, alternately 72 characters and 48 characters long from the left margin.

The PRINT module is organized around the line printer format; each print line generates a pair of teletype lines. Spacing within a line of output on a teletype must take into consideration the wrap-around effect. Vertical spacing options are the same as those for

the line printer. However, the user should be aware that if fewer than 72 characters are to be printed per print line, a space would automatically be created between successive lines of output on the teletype. Hence, by not using print positions 73-120 on any print line, spaces can be inserted between each teletype output line. Horizontal spacing options are also the same as those for the line printer. The character positions of each data field are identified by the relative character position within the printer line; i.e., character positions 2 through 120.

Data Output Suppression

It is possible for the hit file to be in a sort sequence such that a given data value is consecutively repeated in many record sets for a given field. On output, the repetition of this value might be undesirable for a report. Suppression of repetitive values of a data field is accomplished by using a group subphrase following the field title. Valid group subphrases are GROUP n, BREAK n, and G n, where n is a number from 1 to 9. Each application of a group suppression must have a unique identifier n assigned consecutively within the PRINT phrase. For instance, it might be desirable to eliminate the repetitive printing of the values of ORG. This can be achieved using the following PRINT phrase:

```
PRINT ORG G 1, NAME, JOBTITLE, SALARY
```

This would yield the report:

ORG	EMPLOYEE	JOB TITLE	SALARY
2000	LYNN, K. R.	5210	6000
2100	CARR, P. I.	1110	24000
	CALLEGAN, R. E.	5210	5400

Only the first occurrence of each data value is printed for the field title ORG; subsequent contiguous occurrences of the same data value are suppressed.

The precedence of suppression in a report which has more than one group suppression follows the rule that if the value of a data field that is group suppressed changes, all data fields to the right of the given data field in the report will be printed, along with the changed data field. In other words, a change in G 1 will cause any data fields controlled by G 2 through G 9 to be printed even if they themselves have not changed. However, a change in G 4 will cause the data fields controlled by G 5 to G 9 to be printed but will not affect G 1 to G 3. For example, the following PRINT phrase:

```
PRINT ORG G 1, JOBTITLE G 2, SALARY, NAME
```

would yield the following report:

ORG	JOB TITLE	SALARY	EMPLOYEE
2000	5210	6000	LYNN, K. R.
2100	1110	24000	CARR, P. I.
	5210	5400	CALLEGAN, R. E.
2200	5210	6700	MILLS, J. R.
		5900	JOHNS, F. C.
2300	5210	5400	FRANK, E. B.

Even though the value of JOBTITLE did not change for the last four record sets in the example, JOBTITLE was printed whenever ORG changed.

It is possible to control the suppression and printing of a data field by forcing its dependence on the printing of a group suppression field. This is known as keying the suppression of a data field to a group suppression. Keyed suppression of values, not necessarily repetitive values, may be accomplished using the sub-phrases SUPPRESS n, STAR n, and * n, where n is the number of the control group to which the suppressed or controlled data field is keyed. For example, the * 3 data field is suppressed whenever the G 3 field is suppressed. The keyed suppression field must always follow the group suppression field to which it is keyed in the PRINT phrase.

Maximum utility of keyed suppression relies on the user's knowledge of the hit file. For instance, if a hit file were sorted on ORG in ascending order, and SALARY in descending order, it would be possible to list each organization, along with all its employees and only the highest salary within each organization, using keyed suppression:

```
PRINT ORG G 1, NAME, SALARY * 1
```

This PRINT phrase would yield the following report, in which the value of SALARY is only printed when the value of ORG is printed:

ORG	EMPLOYEE	SALARY
2000	LYNN, K. R.	6000
2100	CARR, P. I.	24000
	CALLEGAN, R. E.	
2200	MILLS, J. R.	6700
	JOHNS, F. C.	
2300	FRANK, E. B.	5400

Table Lookup

Data fields which have an argument-function table associated with them in the file dictionary can be converted to the function or output by appending a table lookup sub-phrase to the field title in the PRINT phrase. Valid table lookup sub-phrases are TABLE LOOKUP, TLU, TABLE, LOOKUP, and L. For example, the following PRINT phrase:

```
PRINT ORG NAME, JOBTITLE TLU, SALARY
```

would generate the following report with table lookup functions for JOBTITLE:

ORG	EMPLOYEE	JOB TITLE	SALARY
2000	LYNN, K. R.	SECRETARY	6000
2100	CARR, P. I.	SYSTEMS ENGR	24000
2100	CALLEGAN, R. E.	SECRETARY	5400

If no function exists for a given argument, the argument itself will be output during a table lookup.

Output Volume Control

The number of pages of output in a given report, as well as whether the report is to be a summary report or a full report, may be specified in the PRINT phrase immediately following the word PRINT.

Page Control

The number of pages to which the output of a report is to be truncated to is indicated by a number preceding the first field title. For example, a report of no greater than seven pages is output by the following statement:

```
PRINT 7 NAME, NUM
```

Summary Report

AIDS always assumes that all record sets in the hit file are to be processed for output in a full report, which may also be specified by the subphrases F or FULL following the word PRINT. AIDS also allows summary reports to be output. A summary report is one in which only those record sets in the hit file on which a new variable was computed are processed for output. A summary report must be

specified by the subphrases SUMMARY, SUM, or SM immediately following the word PRINT.

The number of pages to be output for a summary report may be specified by a number following or preceding the summary specification and preceding the first field title. For example, a seven page summary report is indicated by the PRINT phrase:

```
PRINT SUM 7 NAME, NUM, $MONTHLY-SALARY
```

Termination of PRINT Phrase

The PRINT phrase must be terminated by a comma followed by the connector AND. The PRINT phrase may be followed by the phrases REFORMAT, SCRATCH, or PRINT. The user may also go to the operational query commands where he may ACCEPT the query for processing, STORE the query, SAVE the hit file generated by the query, or USE a previously saved hit file as the input data file for this query. Operational query commands are discussed in Section IV.

REFORMAT PHRASE

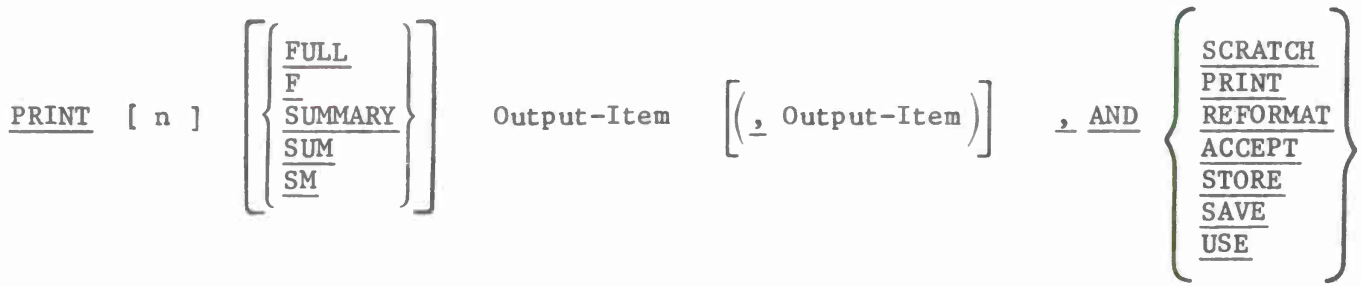
The REFORMAT phrase allows selected data fields within records in a hit file to be extracted and written out onto tape or cards. This subset of the file can then be processed by other user-generated programs and, if a dictionary is written for the output file, it may then be accessed using AIDS.

The basic element of the REFORMAT phrase contains the field title of the source data field, the desired location of the data value in the output record, editing information for the output field such as blank or zero fill, or table lookup conversion. Additional features are also available, such as the ability to insert a constant in the output record either unconditionally or based on the value of the source field and a reference literal. The syntax of the REFORMAT phrase is diagrammed in Figure III-6. The notation used is the same as that used in earlier diagrams.

REFORMAT Output Media

REFORMAT must be the first word of every REFORMAT phrase, followed by a three-character alphanumeric identifier for the output file, and the output media specification, TAPE or CARD:

```
REFORMAT POF CARD,
```



Where Output-Item is:

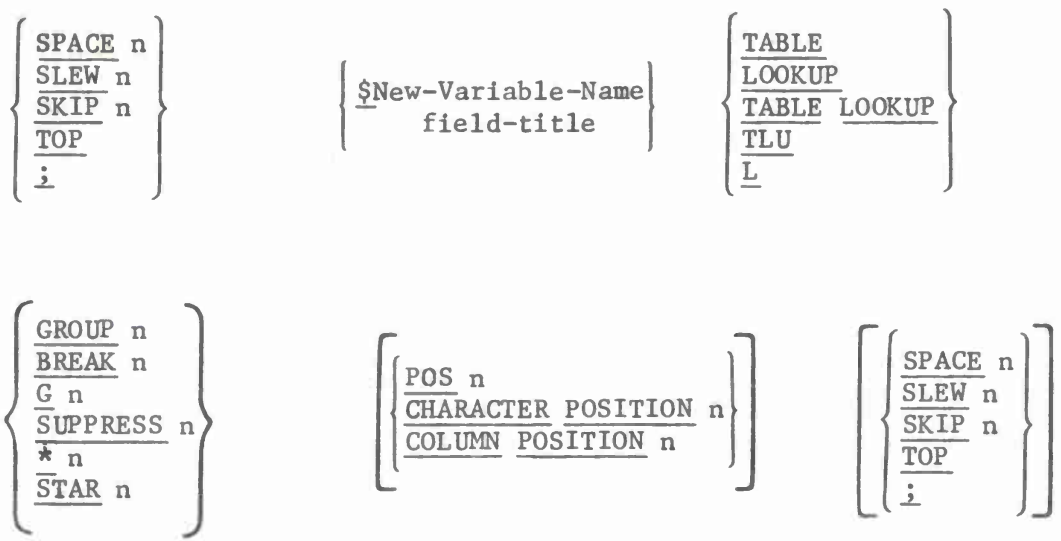


Figure III-5. PRINT Phrase Syntax

This output specification must be separated by a comma from the remainder of the REFORMAT phrase.

The physical output media from the REFORMAT module is always magnetic tape. If CARD has been specified as the output form, the output tape contains card images, which must be converted to punched cards using the G-635 BMC (Bulk Media Conversion) utility program.

Limitations on REFORMAT Output

Limitations on output from the REFORMAT module are as follows:

i. A maximum of 6,000 characters may be output from any record set in the hit file.

ii. The maximum output record size is 1,800 characters.

iii. The maximum number of output data fields for all record types is 400.

Record Definition and Identification

The output media specification must be followed by at least one and not more than 50 record specifications. The word RECORD must be the first word of each of these specifications. Immediately following the word RECORD is a user-assigned numeric between 1 and 50 inclusive which defines each record type. Numbers must be assigned to record types sequentially beginning with 1:

```
RECORD 1
```

Record Size

The phrase SZ n defines the size of the output record in characters, where n is a number between 1 and 1800. For example:

```
RECORD 1 SZ  80
```

When the output media is defined to be CARD, output records are written one per card image, left-justified with trailing blanks.

Break Field

The user must specify the point during hit file processing when the output record is complete and should be written out. This is done by keying the output point to a data value change or break in

the hit file. This break may be keyed to any field in the hit file. The break field in the REFORMAT phrase is normally used either in conjunction with a break field from a COMPUTE phrase, if a computation has been performed in the hit file, or with the order of data which has been sorted in the hit file.

The break field may be keyed to the first or last occurrence of a data value in the hit file using the words BKF and BKL respectively. If a data value is to be written to an output record from every record set, the break field specification of BKF must be used, and the break field must be a valid field title or new variable name in the hit file whose value is unique. For example:

```
RECORD 1 SZ 80 BKF NAME,
```

Reformatted Field Definition

The reformatted field definition follows the reformatted record definition and is separated from it by commas. At least one reformatted field definition must be specified for each record.

Output Field Location

The first parameter in each reformatted field definition is a valid field title or name of a previously defined new variable which specifies the data field that is to be output in the reformatted record. The data field or new variable location in the output record is specified for each output field using the phrase LOC n_1 n_2 where n_1 and n_2 are numbers defining the beginning and ending character positions in which the reformatted field will reside. For example:

```
REFORMAT POF CARD, RECORD 1 SZ 80 BKL SALARY, NAME LOC 3 35,
```

Each reformatted field definition is terminated by a comma. Optional field specifications (described next) must follow the field title and output locations. Optional field specifications may be input in any order following the field location.

Field Justification

An output data value may be right or left justified in an output field using the codes RJ or LJ respectively. For example:

```
NAME LOC 3 35 RJ,
```

If no field justification is specified, the system assumes LJ.

Field Fillers

If the data value does not fill the defined output field location, it may be padded with blanks or zeroes using the codes BF or ZF respectively. For example:

```
NAME LOC 3 35 BF LJ,
```

If no field filler is specified, BF is assumed by the system.

Table Lookup

The use of the code L specifies that the output data value is to be a table lookup function. For example:

```
JOBTITLE LOC 36 45 L BF,
```

The table lookup performed by the REFORMAT module is done using the table associated with the input field title in the dictionary of the input data file.

Zone Stripping

Sign overpunch bits in the first or last character of a data value may be converted to a sign character (+ or -) and moved to the first character position of the output field. This is done using the codes STF or STL respectively. For example:

```
NUM LOC 50 54 ZF STF,

NUM LOC 50 54 ZF STL,
```

An additional character must be allocated to the output field to accommodate the sign character.

Constant Insertion

The constant insertion feature allows the user to specify constants for insertion in an output record, either conditionally or unconditionally. Up to 10 constant insertions may be used in each REFORMAT phrase. The syntax of this feature is as follows:

```
C "Constant" n [Relational "Literal" ]
```

The code C is used to specify that the sub-phrase is a constant insertion sub-phrase. The constant value to be inserted in the reformatted record follows the code C and must be enclosed in quotation marks.

n specifies the starting character position for the constant in the output record. The remainder of the sub-phrase is optional and is used to conditionally insert the constant value in the output record depending on the value of the data field in the REFORMAT phrase which immediately precedes the constant insertion sub-phrase. Constant insertions may not utilize any of the filler, justification, or table lookup options. The constant insertion feature is terminated by a comma.

For example, the user may desire to add the constant data value 2000 in character position 41 of each reformatted record:

```
REFORMAT POF CARD, RECORD 1 SZ 80 BKF NAME, NAME LOC 1 40
LJ BF, C "2000" 41
```

The conditional constant insertion is dependent on the value of the data field specified immediately preceding the constant insertion

```
REFORMAT POF CARD, RECORD 1 SZ 80, BKF NAME, NAME LOC 1 40
LJ BF, C "2000" 41 EQ "CARR, P. I."
```

In this case, the value 2000 is added to the output record starting in character position 41 if the value of the data field NAME is equal to "CARR, P. I." Valid relations are EQ, GT, or LT. Negation is not allowed. The criteria CARR, P. I. must be enclosed in quotes.

REFORMAT Operation

The output record format is dependent upon the sequence in which the data fields for that record are input to the REFORMAT phrase. In general, the last data field moved to a location in the output record, the move being dependent on the input order of the data fields in the REFORMAT phrase, will define what value is in that location. This implies that the REFORMAT operation does not prevent the definition of a data field location which overlaps a location assigned to another data field.

The REFORMAT operation, properly understood, can be useful. To better understand the operation, consider the statement:

```
REFORMAT POF CARD, RECORD 1 SZ 80, $MONTHLY-SALARY LOC 5 16,
$EMP-IN-ORG LOC 1 12, ORG LOC 1 4
```

The new variable $MONTHLY-SALARY is moved to character positions 5 through 6. Assuming that only the right-most four digits of $MONTHLY-SALARY have meaningful data and the remaining leading eight

digits are zeroes, the new variable $EMP-IN-ORG is moved next into positions 1 through 12 over the eight zero digits. Again, assuming that $EMP-IN-ORG contains eight or less digits of value, the data field ORG is moved into the first four character positions of the output record. Obviously, effective use of the REFORMAT operation requires that the user have knowledge of the input file and the data values he wishes to move.

Termination of REFORMAT Phrase

The REFORMAT phrase must be terminated with a comma followed by the connector AND. The REFORMAT phrase may be followed by the phrases SCRATCH or PRINT. The user may also go to the operational query commands where he may ACCEPT the query for processing, STORE the query, SAVE the hit file generated by the query, or USE a previously saved hit file as the input data file for this query. Operational query commands are discussed in Section IV.

SECTION IV

ON-LINE USE

USER STARTUP

The user accesses the General Time-Sharing System (GE-TSS) by following the standard log-on procedures used by his facility. Following the input of a valid user key, GE-TSS requests the name of the system he wishes to access:

```
SYSTEM? AIDS
```

On the same line as the system request, the user inputs the word AIDS, followed by a carriage return to access the AIDS data management system.

At this point in the log-on procedure, AIDS makes a subsystem request followed by a carriage return:

```
SUBSYSTEM?
EXPERT
```

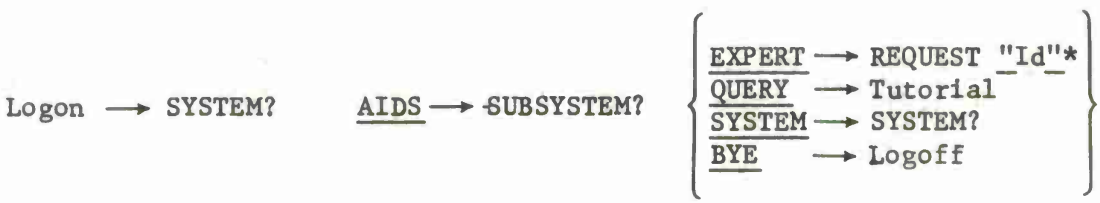
Valid inputs in response to a subsystem request are EXPERT, QUERY, SYSTEM or BYE. The AIDS log-on procedure and the valid inputs for a subsystem request are diagrammed in Figure IV-1. The word EXPERT allows the user to go directly to query input and syntax validation. QUERY allows the user to go into the computer-aided instruction (CAI) mode, where he has the option of two submodes, a read-only mode or a query development mode with step-by-step instruction. The word SYSTEM brings the user up to the system level of GE-TSS, where he may transfer to another system. The word BYE causes GE-TSS to cease operation, and is, in effect, a log-off from GE-TSS.

QUERY FORMULATION

Upon receipt of either the EXPERT or QUERY subsystem requests, AIDS prints the following heading:

```
THIS IS REACTIVE EDIT
AT NN.NNN ON MM/DD/YY
```

where NN.NNN is the clock time, and MM/DD/YY is the current date.



*EXPERT mode options are diagrammed in Figure IV-3.

Figure IV-1. AIDS Logon Procedure

Query formulation may be performed in either the QUERY or EXPERT mode of AIDS. The QUERY mode is not discussed here, as the options available to the user throughout the tutorial are fully outlined. The EXPERT mode is discussed below.

The READY prompting command is output to indicate that user input is expected. Following the initial output heading, AIDS requires the user to input a request identifier, both to identify the transaction, and if desired, to name a query. Valid inputs are shown in Figure IV-2. The word REQUEST must be followed by a transaction (or query) identification which is an alphanumeric code up to a maximum of twelve characters in length enclosed in quotation marks:

```
READY
REQUEST "QUERY-1"
```

Each line of input is terminated by a carriage return, whereupon AIDS performs syntax validation on that line and outputs the next prompting command or an appropriate error message.

The initial request ID may be followed by any of the operational subsystem commands (discussed below) or a query. Each query requires a request ID, but more than one non-query transaction may be performed under a given request ID. The following example illustrates proper input of a query:

```
REQUEST "QUERY-1" RETRIEVE PERSONNEL ORG P, AND PRINT ORG,
READY
```

Upon completion of a query, the user must input an operational query command which direct AIDS as to what is to be done with the query. Operational query commands are discussed in the next subsection of on-line use and are diagrammed in Figure IV-2.

OPERATIONAL COMMANDS

AIDS allows two types of operational commands: operational query commands and operational subsystem commands. Each type is discussed below.

Operational Query Commands

At the completion of a query, the user has the choice of storing the query in the saved query file for subsequent usage, of saving the hit file generated by the query on a specified tape, of using a

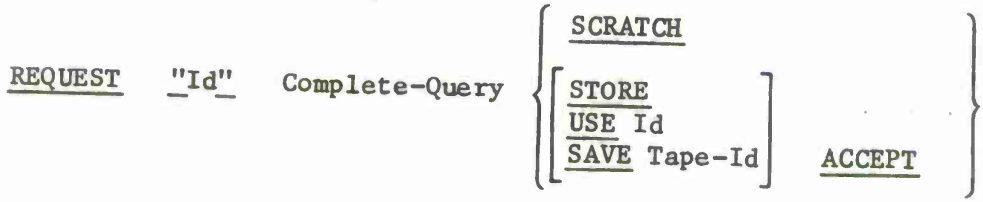


Figure IV-2. Operational Query Commands

designated saved hit file in place of the master data file, of discarding the present query, or of accepting the query for a batch run. The words used for these commands are, respectively, STORE, SAVE, USE, SCRATCH, and ACCEPT. (See Figure IV-2.) The commands to SAVE a hit file, to STORE a query, and to USE a saved hit file are optional. If used, these commands must be followed by an ACCEPT command.

STORE Command

The STORE command is optional and is entered immediately following completion of a query. The STORE command causes the query to be written onto disk in a stored query file so the query may be used again. The command, as is the case with all operational commands, may be entered on the same line as the query, or on the line following a prompting command:

```
REQUEST "QUERY-1" RETRIEVE PERSONNEL ORG P, AND PRINT ORG, AND STORE


REQUEST "QUERY-1" RETRIEVE PERSONNEL ORG P, AND PRINT ORG,
READY
AND STORE
```

USE Command

The USE command is optional and is entered immediately following completion of a query. The USE command identifies a saved hit file to be used as the input data file for this query. The saved hit file name is input immediately following the USE command.

SAVE Command

The SAVE command is optional and is entered immediately following completion of a query. The SAVE command directs the batch processor to save the hit file generated by the query on a designated tape. The five-digit tape number of the tape on which the hit file is to be saved is input immediately following the SAVE command. If no specific tape is desired, the digits "99999" may be entered, and the tape will be assigned by the facility. AIDS cannot identify the reel number for subsequent processing if "99999" is used.

SCRATCH Command

The SCRATCH command is optional and is entered immediately following completion of a query. The SCRATCH command causes the query to be deleted from the system.

ACCEPT Command

The ACCEPT command is entered immediately following completion of a query or following any of the optional operational query commands:

```
REQUEST "QUERY-1" RETRIEVE PERSONNEL ORG P, AND PRINT ORG,
READY
AND STORE ACCEPT
```

The ACCEPT command causes the query to be written onto disk in the query list for subsequent batch processing.

Operational Subsystem Commands

Subsystem commands allow the user to list all stored queries and saved hit files in the system, to delete a stored query from the system, to release a saved hit file and allow the assigned tape to be reused, to request another query input, to return to the system level of GE-TSS, and to terminate dialogue with the system. The words used for these options, respectively, are CATALOG, DELETE, RELEASE, REQUEST, DONE, and STOP. These commands may not be input during active query formulation or preceding the operational query commands. Figure IV-3 illustrates the syntax of the operational query commands.

CATALOG Command

The CATALOG command will list any stored queries and saved hit files for a given user identification.

DELETE Command

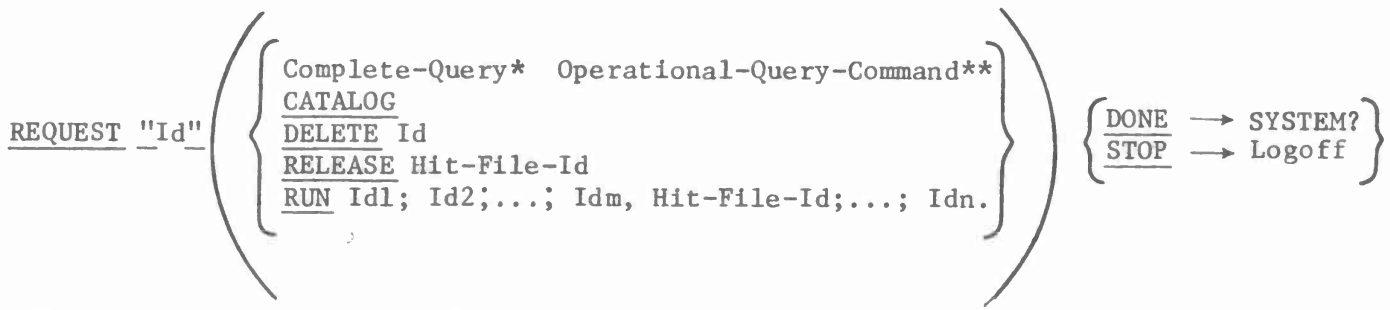
The DELETE command allows the user to delete a stored query from the saved query file. The request ID of the query to be deleted is input following the DELETE command.

RELEASE Command

The RELEASE command allows the user to release a saved hit file from the saved hit file list. The hit file identification is input following the RELEASE command word.

REQUEST Command

The REQUEST command is followed by a request identification, or query ID, enclosed in quotation marks, which names a query and/or indicates to AIDS that a transaction will commence. Each query requires a unique query ID.



*Every query must be preceded by a unique request ID.
**Operational query commands are diagrammed in Figure IV-2.

Figure IV-3. Command Options in EXPERT Mode

RUN Command

The RUN command allows the user to develop requests using stored queries and master files or saved hit files. More than one query, either against a saved hit file or master file, may be input on the line following the RUN command word. The query ID must appear first followed by punctuation. The punctuation may be a comma if a hit file ID is to be input, a semicolon if the query is to be run against the master file and another request follows, or a period if the query is against the master file and there are no further requests. If a comma is found, the next word must identify a saved hit file. The saved hit file ID is followed by a period if there are no further saved query requests or a semicolon if there are further requests.

DONE Command

The DONE command allows the user to return to the system level of GE-TSS, where the user can then request access to another system or terminate his dialogue with GE-TSS.

STOP Command

The STOP command terminates the on-line operation with AIDS. The STOP command is, in effect, a logoff from both AIDS and GE-TSS.

Carolyn A. Marcus

C. A. Marcus
Command and Management Systems

CAM:le

APPENDIX I

PHRASE WORD LIST

Word	Description	Page
RETRIEVE	A data file and selection criteria for retrieval is specified.	32
THROUGH LEVEL n THRU LEVEL n	A restricted level search is specified.	36
EQUAL TO EQUALS EQUAL EQ =	An equals relational is defined.*	38
NE UNEQUAL	An unequals relational is defined.*	38
GREATER THAN GT	A greater than relational is defined.*	38
GE	A greater than or equals relational is defined.*	38
LESS THAN LT	A less than relational is defined.*	38
LE	A less than or equal relational is defined.*	38
PRESENT P	A present relational is defined.	39
I n	An indexed search is specified.	39
M n	A multiple indexed search is specified.	40
U	An unindexed search is specified.	40

*Logical combinations of two of the above relationals are also valid; e.g., LESS THAN OR =. The connectors AND or OR must separate each relational.

Word	Description	Page
SORT	A sorted hit file is output.	44
DSC DESCENDING	A descending sort is requested.	44
ASC ASCENDING	An ascending sort is requested.	44
nn	A report is sorted on the first nn characters of a data field.	44
COMPUTE	A calculated result from an arithmetic formula or a function is output.	46
SUM	A sum of data values are output.	48
COUNT	A count of occurrences of a data field is output.	48
IF	A conditional computation is performed.	49
BREAK	A computation is performed within a sort key.	48
PRINT	A list of individual values is output.	51
SM SUM SUMMARY	A summary report is output.	57
F FULL	A full report is output.	57
TOP	A report is positioned at the top of a page.	53
; SPACE n SLEW n SKIP n	Print lines are spaced within a page.	53
CHARACTER POSITION n COLUMN POSITION n POS n	Data fields are positioned within an output line.	53

Word	Description	Page
TABLE TABLE LOOKUP TLU LOOKUP L	A lookup table function is output.	57
GROUP n BREAK n G n SUPPRESS n * n STAR n	Repeating occurrences of a data value are suppressed for output.	55
REFORMAT	Tape or cards are output.	58
CARD	Card output is requested.	58
TAPE	Tape output is requested.	58
RECORD n	Record type is defined.	60
SZ n	The output record size is defined.	60
BKF BKL	An output record is generated on the first or last occurrence of a value in a data field.	60
LOC n_1 n_2	Location of a field is defined on the output record.	61
BF	Blank fill is defined for an output field.	62
ZF	Zero fill is defined for an output field.	62
LJ	An output field is left-justified.	61
RJ	An output field is right-justified.	61
STF STL	An output field will be dezoned and the appropriate sign will precede the output field (used to strip sign overpunch).	62
L	A table lookup value is output.	62
C	A user-defined constant value is output.	62

APPENDIX II

FILE DEFINITION CARD FORMATS

File Name Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	FILLER	5-9	5	Zero Fill
3	CARD ACTION	10	1	I, M, or D
4	FILLER	11-12	2	Blank Fill
5	FILE NAME	13-48	36	Alphanumeric BCD File Name
6	FILLER	49-51	3	Blank Fill
7	MAX RECORD SIZE	52-57	6	Maximum Words in a Record
8	LEVEL	58-60	3	Number of Levels in File
9	FILLER	61-63	3	Blank Fill
10	SEQ KEY	64-69	6	Information Field
11	FILLER	70-80	11	Blank Fill

Level Structure Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	FILLER	5	1	Zero Fill
3	RECORD ID	6-8	3	Alphanumeric Code for Record
4	CARD TYPE	9	1	Zero Fill
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-48	38	Blank Fill
7	START LOCATION	49-52	4	Starting Character of Record ID Field
8	END LOCATION	53-56	4	Ending Character of Record ID Field
9	FILLER	57-59	3	Blank Fill
10	DICTIONARY LEVEL CODE	60-62	3	Alphanumeric Dictionary Code for Record
11	FILLER	63	1	Blank Fill
12	RECORD SIZE	64-69	6	Record Size in Words
13	FILLER	70-80	11	Blank Fill

Field Definition Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	LEVEL	5	1	Record Level
3	FIELD NUMBER	6-8	3	Numeric Identification of Data Field
4	CARD TYPE	9	1	Always "1"
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-12	2	Blank Fill
7	FIELD NAME	13-48	36	Data Field Name
8	START LOCATION	49-52	4	Starting Character of Data Field
9	END LOCATION	53-56	4	Ending Character of Data Field
10	SEARCH TYPE	57	1	I, M, or U
11	FIELD SIZE	58-59	2	Character Length of Data Field
12	DICTIONARY LEVEL CODE	60-62	3	Dictionary Level Code
13	FILLER	63-80	18	Blank Fill

Report Field Definition Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	LEVEL	5	1	Record Level
3	FIELD NUMBER	6-8	3	Numeric Identification of Data Field
4	CARD TYPE	9	1	Always "2"
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11	1	Blank Fill
7	REPORT FIELD TITLE	12-57	46	Report Field Title
8	FILLER	58-59	2	Blank Fill
9	TITLE CHARACTERS	60-61	2	Character Length of Report Field Title + 3
10	DATA TYPE	62	1	1, 2, or 3
11	DECIMALS IN	63	1	Number of Input Decimal Places
12	DATA FIELD WIDTH	64-66	3	Character Length of Data Field + 3
13	DECIMALS OUT	67	1	Number of Output Decimal Places
14	TABLE LOOKUP CODE	68-69	2	Lookup Table Identification
15	LOOKUP WIDTH	70-72	3	Character Length of Largest Function + 3
16	FILLER	73-80	8	Blank Fill

Table Lookup Card

FIELD	NAME	COLS	SIZE	REMARKS
1	FILE CODE	1-4	4	Alphanumeric Code Name of File
2	CARD TYPE	5	1	Always "9"
3	TABLE NUMBER	6-7	2	Alphanumeric Identification of Table
4	FILLER	8-9	2	Zero Fill
5	CARD ACTION	10	1	I, M, or D
6	FILLER	11-12	2	Blank Fill
7	ARGUMENT	13-24	12	Coded Data Value
8	FUNCTION	25-72	48	Lookup Data Value
9	FILLER	73-80	8	Blank Fill

APPENDIX III

UNIQUE TERMINAL CHARACTERISTICS

GE TERMINET-300

The GE Terminet-300 may be used for remote configurations. During AIDS operation, AIDs generates all output messages printed on this device and controls the processing of all inputs at this device.

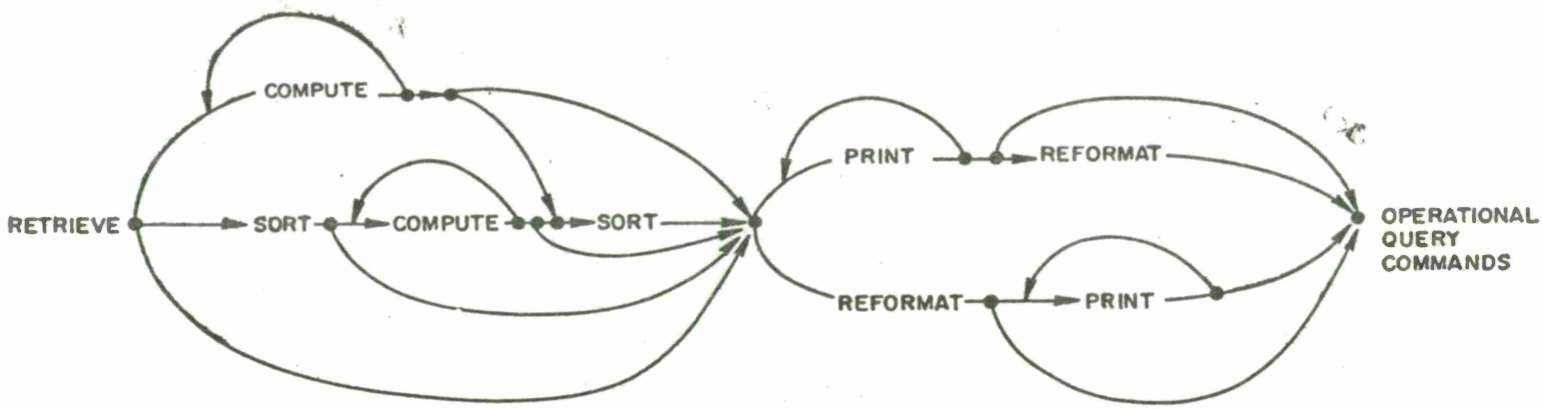
i. Input expected is prompted by a READY command. The user must wait until this is printed; all characters submitted prior to receiving this command are lost.

ii. Input completion is indicated to AIDS by a carriage return.

iii. Backspacing is accomplished by the use of the character @ which AIDS interprets as a backspace; i.e., DAXY@@TA becomes DATA. For each input of the special character @, a character to the left is effectively removed.

iv. Multiple line inputs are allowed. Each input line is terminated by a carriage return, and each new line is preceded by a prompting command.

v. Breaking of the printing of output messages by AIDS is allowed by the use of the interrupt button.



APPENDIX IV VALID ORDER OF PHRASES IN A QUERY

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
The MITRE Corporation P. O. Box 208 Bedford, Massachusetts 01730	UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

Aids Users' Manual

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

C. A. Marcus

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
AUGUST 1971	81	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F19(628)-71-C-0002 b. PROJECT NO. 4060 c. d.	MTR-2203 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Electronic Systems Division, Air Force Systems Command, L. G. Hanscom Field Bedford, Massachusetts 01730

13. ABSTRACT

AIDS is a computer software package designed to provide data management capabilities to a wide variety of users. It is written primarily in Common Business Oriented Language (COBOL) and is designed and implemented on the Honeywell G-635. This system, originally developed for NASA by General Electric Co., Apollo Systems Division, was named Manned Space Flight-Data Processing System (MSF-DPS). Modifications were made to improve the capabilities of MSF-DPS. These modifications, designed to meet interim requirements of the Air Force Data Services Center (AF/ACS), provide a responsive, versatile data management system for users of Honeywell (GE) 600 series computers.

This technical reports, designed for users of AIDS, details the features of this system and provides examples of its use. Detailed system description (installation, maintenance, internal linkages, etc.) are not contained in this report. This information is contained in the AIDS Operations Manual (General Electric Co., "AIDS Version Description Document," July 1971). Organizations using AIDS may further modify or enhance this system independently of ESD; therefore, no attempt will be made by ESD to update this Technical Report if such changes are made.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
SOFTWARE						
DATA MANAGEMENT						
COMPUTER						
COBOL						
QUERY LANGUAGE						
CAI						
TAPE FILES						
G635						
GE-TSS						